

EXHIBIT C



US010477994B2

(12) **United States Patent**
Johnson

(10) **Patent No.:** **US 10,477,994 B2**

(45) **Date of Patent:** **Nov. 19, 2019**

(54) **SYSTEM AND METHOD FOR LOCATION
BASED EXCHANGES OF DATA
FACILITIATING DISTRIBUTED
LOCATIONAL APPLICATIONS**

(52) **U.S. Cl.**
CPC **A47G 1/16** (2013.01); **F16B 13/00**
(2013.01); **F16B 45/00** (2013.01)

(58) **Field of Classification Search**

CPC H04W 4/02; H04W 64/00; H04W 4/023;
H04W 12/06; H04W 40/20; H04W
40/244; H04W 92/18; H04W 4/21; H04W
4/50; H04W 4/80; H04W 68/005; H04W
88/02

See application file for complete search history.

(71) Applicant: **William J. Johnson**, Flower Mound,
TX (US)

(72) Inventor: **William J. Johnson**, Flower Mound,
TX (US)

(*) Notice: Subject to any disclaimer, the term of this
patent is extended or adjusted under 35
U.S.C. 154(b) by 0 days.

(56) **References Cited**

U.S. PATENT DOCUMENTS

626,615 A 6/1899 Hood
4,021,780 A 5/1977 Narey et al.
(Continued)

(21) Appl. No.: **16/375,836**

(22) Filed: **Apr. 4, 2019**

(65) **Prior Publication Data**

US 2019/0231097 A1 Aug. 1, 2019

FOREIGN PATENT DOCUMENTS

EP 0712227 5/1996
EP 915590 5/1999
(Continued)

Related U.S. Application Data

(63) Continuation of application No. 16/147,532, filed on
Sep. 28, 2018, now Pat. No. 10,292,011, which is a
continuation of application No. 15/218,039, filed on
Jul. 24, 2016, now Pat. No. 10,111,034, which is a
continuation of application No. 14/752,945, filed on
Jun. 28, 2015, now Pat. No. 9,456,303, which is a
continuation of application No. 13/972,125, filed on
Aug. 21, 2013, now Pat. No. 9,078,095, which is a
continuation of application No. 12/590,831, filed on
Nov. 13, 2009, now Pat. No. 8,634,796, which is a
continuation-in-part of application No. 12/287,064,
(Continued)

OTHER PUBLICATIONS

Bill N. Schilit and Marvin M. Theimer, Disseminating Active Map
Information Mobile Hosts, IEEE Network, Sep./Oct. 1994.

(Continued)

Primary Examiner — Liton Miah

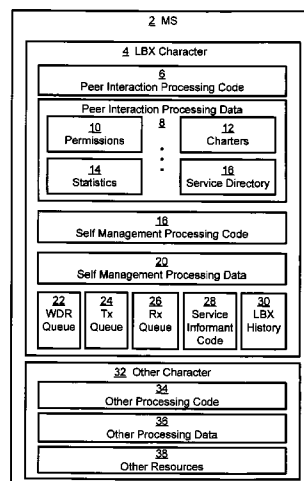
(74) *Attorney, Agent, or Firm* — Yudell Isidore PLLC

(57) **ABSTRACT**

Mobile data processing Systems (MSs) interact with systems
in their vicinity, and with each other, in communications and
interoperability. Information transmitted inbound to, trans-
mitted outbound from, is in process at, or is application
modified at a mobile data processing system triggers pro-
cessing of actions in accordance with user configurations,
for example to present content to a user.

19 Claims, 322 Drawing Sheets

(51) **Int. Cl.**
H04M 11/04 (2006.01)
A47G 1/16 (2006.01)
F16B 13/00 (2006.01)
F16B 45/00 (2006.01)



US 10,477,994 B2

Page 2

Related U.S. Application Data

filed on Oct. 3, 2008, now Pat. No. 8,639,267, which is a continuation-in-part of application No. 12/077,041, filed on Mar. 14, 2008, now Pat. No. 8,600,341.

(56)

References Cited

U.S. PATENT DOCUMENTS

4,255,619 A	3/1981	Saito	5,625,668 A	4/1997	Loomis
4,445,118 A	4/1984	Taylor et al.	5,627,549 A	5/1997	Park
4,536,647 A	8/1985	Atalla et al.	5,636,245 A	6/1997	Ernst et al.
4,644,351 A	2/1987	Zabarsky et al.	5,646,632 A	7/1997	Khan et al.
4,757,267 A	7/1988	Riskin	5,654,959 A	8/1997	Baker et al.
4,841,560 A	6/1989	Chan et al.	5,657,375 A	8/1997	Connolly et al.
4,845,504 A	7/1989	Roberts et al.	5,661,492 A	8/1997	Shoap et al.
4,922,516 A	5/1990	Butler et al.	5,663,734 A	9/1997	Krasner
4,973,952 A	11/1990	Malec et al.	5,664,948 A	9/1997	Dimitriadis et al.
4,974,170 A	11/1990	Bouve et al.	5,666,481 A	9/1997	Lewis
4,977,399 A	12/1990	Price et al.	5,677,905 A	10/1997	Bigham
5,089,814 A	2/1992	DeLuca et al.	5,687,212 A	11/1997	Kinser, Jr. et al.
5,095,532 A	3/1992	Mardus	5,689,431 A	11/1997	Rudow et al.
5,121,126 A	6/1992	Clagett	5,694,453 A	12/1997	Fuller et al.
5,122,795 A	6/1992	Cubley et al.	5,701,301 A	12/1997	Weisser, Jr.
5,131,020 A	7/1992	Liebesny et al.	5,704,049 A	12/1997	Briechele
5,185,857 A	2/1993	Rozmanith et al.	5,712,899 A	1/1998	Pace, II
5,196,031 A	3/1993	Ordish	5,713,075 A	1/1998	Threadgill et al.
5,214,793 A	5/1993	Conway et al.	5,714,948 A	2/1998	Farmakis et al.
5,223,844 A	6/1993	Mansell et al.	5,717,688 A	2/1998	Belanger et al.
5,243,652 A	9/1993	Teare et al.	5,720,033 A	2/1998	Deo
5,245,608 A	9/1993	Deaton et al.	5,724,521 A	3/1998	Dedrick
5,264,822 A	11/1993	Vogelman et al.	5,727,057 A	3/1998	Emery et al.
5,265,070 A	11/1993	Minowa	5,729,680 A	3/1998	Belanger et al.
5,303,393 A	4/1994	Noreen et al.	5,771,283 A	6/1998	Chang et al.
5,321,242 A	6/1994	Heath, Jr.	5,774,534 A	6/1998	Mayer
5,337,044 A	8/1994	Folger et al.	5,778,304 A	7/1998	Grube et al.
5,347,632 A	9/1994	Filepp et al.	5,790,974 A	8/1998	Tognazzini
5,363,245 A	11/1994	Borello	5,794,210 A	8/1998	Goldhaber et al.
5,363,377 A	11/1994	Sharpe	5,796,727 A	8/1998	Harrison et al.
5,365,516 A	11/1994	Jandrell	5,798,733 A	8/1998	Ethridge
5,371,794 A	12/1994	Diffie et al.	5,806,018 A	9/1998	Smith et al.
5,390,237 A	2/1995	Hoffman et al.	5,812,763 A	9/1998	Teng
5,404,505 A	4/1995	Levinson	5,819,155 A	10/1998	Worthey et al.
5,432,841 A	7/1995	Rimer	5,826,195 A	10/1998	Westerlage et al.
5,444,444 A	8/1995	Ross	5,758,049 A	11/1998	Johnson et al.
5,451,757 A	9/1995	Heath, Jr.	5,835,061 A	11/1998	Stewart
5,455,807 A	10/1995	Nepple	5,838,774 A	11/1998	Weisser, Jr.
5,461,627 A	10/1995	Rypinski	5,842,010 A	11/1998	Jain et al.
5,469,362 A	11/1995	Hunt et al.	5,845,211 A	12/1998	Roach
5,475,735 A	12/1995	Williams et al.	5,852,775 A	12/1998	Hidary
5,485,163 A	1/1996	Singer et al.	5,855,007 A	12/1998	Jovicic et al.
5,487,103 A	1/1996	Richardson	5,870,555 A	2/1999	Pruett et al.
5,493,309 A	2/1996	Bjornholt et al.	5,870,724 A	2/1999	Lawlor et al.
5,497,414 A	3/1996	Bartholomew	5,875,186 A	2/1999	Belanger et al.
5,504,482 A	4/1996	Schreder	5,875,401 A	2/1999	Rochkind
5,511,111 A	4/1996	Serbetcioglu et al.	5,878,126 A	3/1999	Velamuri et al.
5,511,233 A	4/1996	Otten	5,880,958 A	3/1999	Helms et al.
5,512,908 A	4/1996	Herrick	5,881,131 A	3/1999	Farris et al.
5,513,263 A	4/1996	White et al.	5,884,284 A	3/1999	Peters et al.
5,528,248 A	6/1996	Steiner et al.	5,887,259 A	3/1999	Zicker et al.
5,539,395 A	7/1996	Buss et al.	5,889,953 A	3/1999	Thebaut et al.
5,544,354 A	8/1996	May et al.	5,892,454 A	4/1999	Schipper et al.
5,559,520 A	9/1996	Barzegar et al.	5,896,440 A	4/1999	Reed et al.
5,561,704 A	10/1996	Samilando	5,897,640 A	4/1999	Veghte et al.
5,566,235 A	10/1996	Hetz	5,903,636 A	5/1999	Malik
5,581,479 A	12/1996	McLaughlin	5,907,544 A	5/1999	Rypinski
5,583,864 A	12/1996	Lightfoot et al.	5,920,846 A	7/1999	Storch et al.
5,586,254 A	12/1996	Kondo et al.	5,922,040 A	7/1999	Prabhakaran
5,588,042 A	12/1996	Comer	5,923,702 A	7/1999	Brenner et al.
5,590,196 A	12/1996	Moreau	5,933,420 A	8/1999	Jaszewski et al.
5,590,398 A	12/1996	Matthews	5,938,721 A	8/1999	Dussell et al.
5,592,470 A	1/1997	Rudrapatna et al.	5,949,867 A	9/1999	Sonnenberg
5,594,779 A	1/1997	Goodman	5,950,130 A	9/1999	Coursey
5,596,625 A	1/1997	LeBlanc	5,961,593 A	10/1999	Gabber et al.
5,602,843 A	2/1997	Gray	5,963,866 A	10/1999	Palamara et al.
5,608,854 A	3/1997	Labedz et al.	5,963,913 A	10/1999	Henneuse et al.
5,610,973 A	3/1997	Comer	5,968,176 A	10/1999	Nessett et al.
5,625,364 A	4/1997	Herrick et al.	5,969,678 A	10/1999	Stewart
			5,982,867 A	11/1999	Urban et al.
			5,983,091 A	11/1999	Rodriguez
			5,987,381 A	11/1999	Oshizawa
			5,991,287 A	11/1999	Diepstraten et al.
			5,995,015 A	11/1999	DeTemple et al.
			6,006,090 A	12/1999	Coleman et al.
			6,009,398 A	12/1999	Mueller et al.
			6,011,975 A	1/2000	Emery et al.
			6,018,293 A	1/2000	Smith et al.
			6,026,151 A	2/2000	Bauer et al.

US 10,477,994 B2

Page 3

(56)

References Cited

U.S. PATENT DOCUMENTS

6,028,921	A	2/2000	Malik et al.	6,385,591	B1	5/2002	Mankoff
6,047,327	A	4/2000	Tso et al.	6,389,055	B1	5/2002	August et al.
6,055,637	A	4/2000	Hudson et al.	6,389,426	B1	5/2002	Turnbull et al.
6,058,106	A	5/2000	Cudak et al.	6,393,482	B1	5/2002	Rai et al.
6,067,082	A	5/2000	Enmei	6,400,722	B1	6/2002	Chuah et al.
6,067,297	A	5/2000	Beach	6,405,123	B1	6/2002	Rennard et al.
6,073,062	A	6/2000	Hoshino et al.	6,407,673	B1	6/2002	Lane
6,076,080	A	6/2000	Morscheck et al.	6,408,307	B1	6/2002	Semple et al.
6,085,086	A	7/2000	La Porta et al.	6,414,635	B1	7/2002	Stewart et al.
6,091,956	A	7/2000	Hollenberg	6,414,950	B1	7/2002	Rai et al.
6,101,381	A	8/2000	Tajima et al.	6,415,019	B1	7/2002	Savaglio et al.
6,101,443	A	8/2000	Kato et al.	6,418,308	B1	7/2002	Heinonen et al.
6,112,186	A	8/2000	Bergh et al.	6,421,441	B1	7/2002	Dzuban
6,115,669	A	9/2000	Watanabe et al.	6,421,714	B1	7/2002	Rai et al.
6,122,520	A	9/2000	Want et al.	6,427,073	B1	7/2002	Kortelsalmi et al.
6,133,853	A	10/2000	Obradovich et al.	6,427,115	B1	7/2002	Sekiyama
6,138,003	A	10/2000	Kingdon et al.	6,427,119	B1	7/2002	Stefan et al.
6,138,119	A	10/2000	Hall et al.	6,430,276	B1	8/2002	Bouvier et al.
6,141,609	A	10/2000	Herdeg et al.	6,430,562	B1	8/2002	Kardos et al.
6,144,645	A	11/2000	Struhsaker et al.	6,442,391	B1	8/2002	Johansson et al.
6,154,152	A	11/2000	Ito	6,442,479	B1	8/2002	Barton
6,154,637	A	11/2000	Wright et al.	6,442,687	B1	8/2002	Savage
6,157,829	A	12/2000	Grube et al.	6,449,272	B1	9/2002	Chuah et al.
6,157,946	A	12/2000	Itakura et al.	6,449,497	B1	9/2002	Kirbas et al.
6,163,274	A	12/2000	Lindgren	6,452,498	B2	9/2002	Stewart
6,167,255	A	12/2000	Kennedy, III et al.	6,456,234	B1	9/2002	Johnson
6,182,226	B1	1/2001	Reid et al.	6,463,533	B1	10/2002	Calamera et al.
6,184,829	B1	2/2001	Stilp	6,470,378	B1	10/2002	Tracton et al.
6,185,426	B1	2/2001	Alperovich et al.	6,470,447	B1	10/2002	Lambert et al.
6,185,484	B1	2/2001	Rhinehart	6,473,626	B1	10/2002	Nevoux et al.
6,192,314	B1	2/2001	Khavakh et al.	6,477,382	B1	11/2002	Mansfield et al.
6,202,054	B1	3/2001	Lawlor et al.	6,477,526	B2	11/2002	Hayashi et al.
6,205,478	B1	3/2001	Sugano et al.	6,484,029	B2	11/2002	Hughes et al.
6,208,854	B1	3/2001	Roberts et al.	6,484,092	B2	11/2002	Seibel
6,208,866	B1	3/2001	Rouhollahzadeh et al.	6,484,148	B1	11/2002	Boyd
6,226,277	B1	5/2001	Chuah	6,490,291	B1	12/2002	Lee et al.
6,229,477	B1	5/2001	Chang et al.	6,496,491	B2	12/2002	Chuah et al.
6,229,810	B1	5/2001	Gerszberg et al.	6,496,931	B1	12/2002	Rajchel et al.
6,233,329	B1	5/2001	Urban et al.	6,505,046	B1	1/2003	Baker
6,233,452	B1	5/2001	Nishino	6,505,048	B1	1/2003	Moles et al.
6,236,360	B1	5/2001	Rudow et al.	6,505,049	B1	1/2003	Dorenbosch
6,236,362	B1	5/2001	Leblanc et al.	6,505,120	B2	1/2003	Yamashita et al.
6,236,940	B1	5/2001	Rudow et al.	6,505,163	B1	1/2003	Zhang et al.
6,246,361	B1	6/2001	Weill et al.	6,512,754	B2	1/2003	Feder et al.
6,246,948	B1	6/2001	Thakker	6,516,055	B1	2/2003	Bedeski et al.
6,252,544	B1	6/2001	Hoffberg	6,516,416	B2	2/2003	Gregg et al.
6,259,405	B1	7/2001	Stewart et al.	6,519,252	B2	2/2003	Sallberg
6,263,209	B1	7/2001	Reed et al.	6,519,458	B2	2/2003	Oh et al.
6,278,938	B1	8/2001	Alumbaugh	6,522,876	B1	2/2003	Weiland et al.
6,285,665	B1	9/2001	Chuah et al.	6,526,275	B1	2/2003	Calvert
6,285,931	B1	9/2001	Hattori et al.	6,526,349	B2	2/2003	Bullock et al.
6,298,234	B1	10/2001	Brunner	6,532,418	B2	3/2003	Chun et al.
6,308,273	B1	10/2001	Goertzel et al.	6,545,596	B1	4/2003	Moon
6,311,069	B1	10/2001	Havinis et al.	6,546,257	B1	4/2003	Stewart
6,317,718	B1	11/2001	Fano	6,560,442	B1	5/2003	Yost et al.
6,321,092	B1	11/2001	Fitch et al.	6,560,461	B1	5/2003	Fomukong et al.
6,324,396	B1	11/2001	Vasa et al.	6,571,279	B1	5/2003	Herz et al.
6,326,918	B1	12/2001	Stewart	6,577,643	B1	6/2003	Rai et al.
6,327,254	B1	12/2001	Chuah	6,577,644	B1	6/2003	Chuah et al.
6,327,357	B1	12/2001	Meek et al.	6,594,482	B1	7/2003	Findikli et al.
6,332,127	B1	12/2001	Bandera et al.	6,615,131	B1	9/2003	Rennard et al.
6,332,163	B1	12/2001	Bowman-Amuah	6,618,474	B1	9/2003	Reese
6,340,958	B1	1/2002	Cantu et al.	6,618,593	B1	9/2003	Drutman et al.
6,343,290	B1	1/2002	Cossins et al.	6,622,016	B1	9/2003	Sladek et al.
6,345,288	B1	2/2002	Reed et al.	6,628,627	B1	9/2003	Zendle et al.
6,353,664	B1	3/2002	Cannon et al.	6,628,928	B1	9/2003	Crosby et al.
6,359,880	B1	3/2002	Curry et al.	6,628,938	B1	9/2003	Rachabathuni et al.
6,360,101	B1	3/2002	Irvin	6,633,633	B1	10/2003	Bedingfield
6,366,561	B1	4/2002	Bender	6,640,184	B1	10/2003	Rabe
6,370,389	B1	4/2002	Isomursu et al.	6,647,257	B2	11/2003	Owensby
6,377,548	B1	4/2002	Chuah et al.	6,647,269	B2	11/2003	Hendrey et al.
6,377,810	B1	4/2002	Geiger et al.	6,650,901	B1	11/2003	Schuster et al.
6,377,982	B1	4/2002	Rai et al.	6,654,610	B1	11/2003	Chen et al.
6,381,311	B2	4/2002	Joyce et al.	6,662,014	B1	12/2003	Walsh
6,385,531	B2	5/2002	Bates et al.	6,665,536	B1	12/2003	Mahany
				6,665,718	B1	12/2003	Chuah et al.
				6,671,272	B2	12/2003	Vaziri et al.
				6,675,017	B1	1/2004	Zellner et al.
				6,675,208	B1	1/2004	Rai et al.

US 10,477,994 B2

Page 4

(56)

References Cited

U.S. PATENT DOCUMENTS

6,677,894 B2	1/2004	Sheynblat et al.	2001/0034709 A1	10/2001	Stoifo et al.
6,697,018 B2	2/2004	Stewart et al.	2001/0049275 A1	12/2001	Pierry et al.
6,697,783 B1	2/2004	Brinkman et al.	2001/0051911 A1	12/2001	Marks et al.
6,701,160 B1	3/2004	Pinder et al.	2002/0035474 A1	3/2002	Alpdemir
6,701,251 B2	3/2004	Stefan et al.	2002/0035493 A1	3/2002	Mozayeny et al.
6,704,311 B1	3/2004	Chuah et al.	2002/0037709 A1	3/2002	Bhatia et al.
6,716,101 B1	4/2004	Meadows et al.	2002/0037722 A1	3/2002	Hussain et al.
6,721,406 B1	4/2004	Contractor	2002/0037731 A1	3/2002	Mao et al.
6,725,048 B2	4/2004	Mao et al.	2002/0037744 A1	3/2002	Bhatia et al.
6,731,238 B2	5/2004	Johnson	2002/0037750 A1	3/2002	Hussain et al.
6,732,080 B1	5/2004	Blants	2002/0038362 A1	3/2002	Bhatia et al.
6,732,101 B1	5/2004	Cook	2002/0038384 A1	3/2002	Khan et al.
6,732,176 B1	5/2004	Stewart et al.	2002/0038386 A1	3/2002	Bhatia et al.
6,738,808 B1	5/2004	Zellner et al.	2002/0046069 A1	4/2002	Mozayeny et al.
6,754,504 B1	6/2004	Reed	2002/0046077 A1	4/2002	Mozayeny et al.
6,754,582 B1	6/2004	Smith et al.	2002/0046090 A1	4/2002	Stewart
6,759,960 B2	7/2004	Stewart et al.	2002/0052781 A1	5/2002	Aufricht et al.
6,772,064 B1	8/2004	Smith et al.	2002/0077083 A1	6/2002	Zellner et al.
6,799,049 B1	9/2004	Zellner et al.	2002/0077084 A1	6/2002	Zellner et al.
6,801,509 B1	10/2004	Chuah et al.	2002/0077118 A1	6/2002	Zellner et al.
6,816,720 B2	11/2004	Hussain et al.	2002/0077130 A1	6/2002	Owensby
6,819,929 B2	11/2004	Antonucci et al.	2002/0077897 A1	6/2002	Zellner et al.
6,820,062 B1	11/2004	Gupta et al.	2002/0087335 A1	7/2002	Meyers et al.
6,829,475 B1	12/2004	Lee et al.	2002/0090932 A1	7/2002	Bhatia et al.
6,850,758 B1	2/2005	Paul et al.	2002/0091991 A1	7/2002	Castro
6,867,733 B2	3/2005	Sandhu et al.	2002/0095312 A1	7/2002	Wheat
6,868,074 B1	3/2005	Hanson	2002/0095454 A1	7/2002	Reed et al.
6,874,011 B1	3/2005	Spielman	2002/0102993 A1	8/2002	Hendrey et al.
6,876,858 B1	4/2005	Duvall et al.	2002/0107027 A1	8/2002	O'Neil
6,898,569 B1	5/2005	Bansal et al.	2002/0120713 A1	8/2002	Gupta et al.
6,937,869 B1	8/2005	Rayburn	2002/0161637 A1	10/2002	Sugaya
6,937,998 B1	8/2005	Swartz et al.	2002/0174147 A1	11/2002	Wang et al.
6,954,147 B1	10/2005	Cromer et al.	2003/0003990 A1	1/2003	Von Kohorn
6,985,747 B2	1/2006	Chithambaram	2003/0016233 A1	1/2003	Charpentier
6,999,572 B1	2/2006	Shaffer et al.	2003/0018527 A1	1/2003	Filepp et al.
7,005,985 B1	2/2006	Steeves	2003/0030731 A1	2/2003	Colby
7,009,556 B2	3/2006	Stewart et al.	2003/0140088 A1	7/2003	Robinson et al.
7,023,995 B2	4/2006	Olsson	2003/0169151 A1	9/2003	Ebling et al.
7,043,231 B2	5/2006	Bhatia et al.	2004/0002329 A1	1/2004	Bhatia et al.
7,058,594 B2	6/2006	Stewart et al.	2004/0097243 A1	5/2004	Zellner et al.
7,069,319 B2	6/2006	Zellner et al.	2004/0111269 A1	6/2004	Koch
7,085,555 B2	8/2006	Zellner et al.	2004/0116131 A1	6/2004	Hochrainer et al.
7,103,368 B2	9/2006	Teshima	2004/0151151 A1	8/2004	Kubler et al.
7,103,476 B2	9/2006	Smith et al.	2004/0164898 A1	8/2004	Stewart
7,106,843 B1	9/2006	Gainsboro et al.	2004/0186902 A1	9/2004	Stewart et al.
7,110,749 B2	9/2006	Zellner et al.	2004/0201459 A1	10/2004	Rich et al.
7,116,977 B1	10/2006	Moton et al.	2004/0203903 A1	10/2004	Wilson et al.
7,124,101 B1	10/2006	Mikurak	2004/0205198 A1	10/2004	Zellner et al.
7,130,631 B2	10/2006	Enzmann et al.	2004/0228330 A1	11/2004	Kubler et al.
7,139,722 B2	11/2006	Perrella et al.	2004/0246940 A1	12/2004	Kubler et al.
7,155,199 B2	12/2006	Zalewski et al.	2004/0252051 A1	12/2004	Johnson
7,177,651 B1	2/2007	Almassy	2004/0264442 A1	12/2004	Kubler et al.
7,181,225 B1	2/2007	Moton et al.	2004/0266453 A1	12/2004	Maanoja et al.
7,181,529 B2	2/2007	Bhatia et al.	2005/0002419 A1	1/2005	Doviak et al.
7,188,027 B2	3/2007	Smith et al.	2005/0004838 A1	1/2005	Perkowski et al.
7,190,960 B2	3/2007	Wilson et al.	2005/0017068 A1	1/2005	Zalewski et al.
7,203,502 B2	4/2007	Wilson et al.	2005/0043036 A1	2/2005	Ioppe et al.
7,212,829 B1	5/2007	Lau et al.	2005/0050227 A1	3/2005	Michelman
7,224,978 B2	5/2007	Zellner et al.	2005/0060365 A1	3/2005	Robinson et al.
7,236,799 B2	6/2007	Wilson et al.	2005/0096067 A1	5/2005	Martin
RE39,717 E	7/2007	Yates et al.	2005/0114777 A1	5/2005	Szeto
7,245,925 B2	7/2007	Zellner	2005/0151655 A1	7/2005	Hamrick et al.
7,260,378 B2	8/2007	Holland et al.	2005/0246097 A1	11/2005	Hamrick et al.
7,272,493 B1	9/2007	Hamrick et al.	2005/0272445 A1	12/2005	Zellner
7,292,939 B2	11/2007	Smith et al.	2005/0283833 A1	12/2005	Lalonde et al.
7,295,924 B2	11/2007	Smith et al.	2006/0009190 A1	1/2006	Laliberte
7,362,851 B2	4/2008	Contractor	2006/0010202 A1	1/2006	Blackstock et al.
7,383,052 B2	6/2008	Moton et al.	2006/0022048 A1	2/2006	Johnson
7,386,396 B2	6/2008	Johnson	2006/0030335 A1	2/2006	Zellner et al.
7,787,887 B2	8/2010	Gupta et al.	2006/0030339 A1	2/2006	Zhovnirovsky et al.
2001/0001239 A1	5/2001	Stewart	2006/0059043 A1	3/2006	Chan et al.
2001/0005864 A1	6/2001	Mousseau et al.	2006/0089134 A1	4/2006	Moton et al.
2001/0007450 A1	7/2001	Begum	2006/0094447 A1	5/2006	Zellner
2001/0021646 A1	9/2001	Antonucci et al.	2006/0099966 A1	5/2006	Moton et al.
2001/0028301 A1	10/2001	Geiger et al.	2006/0105784 A1	5/2006	Zellner et al.
			2006/0106537 A1	5/2006	Hamrick et al.
			2006/0136544 A1	6/2006	Atsmon et al.
			2006/0164302 A1	7/2006	Stewart et al.
			2006/0167986 A1	7/2006	Trzyna et al.

US 10,477,994 B2

Page 5

(56) **References Cited**
U.S. PATENT DOCUMENTS

2006/0183467	A1	8/2006	Stewart et al.
2006/0189327	A1	8/2006	Zellner et al.
2006/0189332	A1	8/2006	Benco et al.
2006/0194589	A1	8/2006	Sankisa
2006/0195570	A1	8/2006	Zellner et al.
2006/0198359	A1	9/2006	Fok et al.
2006/0240828	A1	10/2006	Jain et al.
2006/0252465	A1	11/2006	Karstens et al.
2006/0253252	A1	11/2006	Hamrick et al.
2007/0005188	A1	1/2007	Johnson
2007/0010260	A1	1/2007	Zellner et al.
2007/0042789	A1	2/2007	Moton et al.
2007/0105565	A1	5/2007	Enzmann et al.
2007/0124721	A1	5/2007	Cowing et al.
2007/0136603	A1	6/2007	Kuecukyan
2007/0232326	A1	10/2007	Johnson
2007/0233387	A1	10/2007	Johnson
2007/0244633	A1	10/2007	Phillips et al.
2007/0250920	A1	10/2007	Lindsay
2007/0275730	A1	11/2007	Bienas et al.
2007/0276587	A1	11/2007	Johnson
2007/0281716	A1	12/2007	Altman et al.
2007/0287473	A1	12/2007	Dupray
2008/0030308	A1	2/2008	Johnson
2008/0071761	A1	3/2008	Singh et al.
2008/0096529	A1	4/2008	Zellner
2008/0170679	A1	7/2008	Sheha et al.
2008/0301561	A1	12/2008	Bain
2009/0054077	A1	2/2009	Gauthier et al.
2009/0067593	A1	3/2009	Ahlin
2009/0167524	A1	7/2009	Chesnutt et al.
2009/0190734	A1	7/2009	White et al.
2009/0233622	A1	9/2009	Johnson
2009/0233623	A1	9/2009	Johnson et al.
2009/0233633	A1	9/2009	Johnson
2010/0069035	A1	3/2010	Johnson
2010/0146160	A1	6/2010	Piekarski
2010/0159946	A1	6/2010	Cheung et al.
2010/0227595	A1	9/2010	Johnson
2010/0235748	A1	9/2010	Johnson et al.
2011/0021145	A1	1/2011	Johnson et al.

FOREIGN PATENT DOCUMENTS

EP	917320	5/1999
EP	935364	8/1999
EP	924914	4/2003
EP	779752	6/2004
EP	1435749	7/2004
EP	1445923	8/2004
EP	838933	4/2008
GB	2396779	6/2004
JP	01-194628	8/1989
JP	03-128540	5/1991
JP	07-234789	9/1995
JP	07-288514	10/1995
JP	07-319706	12/1995
JP	08-44568	2/1996
JP	08-87296	4/1996
JP	11-168478	6/1999
WO	WO 98/19484	5/1998

WO	WO 99/16263	4/1999
WO	WO 99/27716	6/1999
WO	WO 99/51005	10/1999
WO	WO 99/55012	10/1999
WO	WO 00/02365	1/2000
WO	WO 00/076249	12/2000
WO	WO 02/11407	2/2002
WO	WO 04/80092	9/2004

OTHER PUBLICATIONS

Andy Harter and Andy Hooper, A Distributed Location system for the Active Office, IEEE Network, Jan./Feb. 1994.

Max J. Egenhofer, Spatial SQL: A Query and Presentation Language, IEEE Network, Feb. 1994.

Mike Spreitzer and Marvin Theimer, Providing Location Information in a Ubiquitous Computing Environment, Proceedings of the Fourteenth ACM Symposium on Operating Systems Principles, Dec. 1993.

George W. Fitzmaurice, Situated Information Spaces and Spatially Aware Palmtop Computers, Communication of the ACM, Jul. 1993.

Ronald Azuma, Tracking Requirements for Augmented Reality, Communications of the ACM, vol. 36 No. 1, Jan. 1992.

Roy Want, et al., The Active Badge Location System, ACM Transactions on Information Systems, vol. 10, No. 1, Jan. 1992.

Marvin White, Emerging Requirements for Digital Maps for In-Vehicle Pathfinding and Other Traveller Assistance, Vehicular Navigation and Information Systems Conference Proceedings, Part 1, Oct. 1991.

Fred Phail, The Power of a Personal Computer for Car Information and Communications Systems, Vehicular Navigation and Information Systems Conference Proceedings, Part 1, Oct. 1991.

Thomas A. Dingus, et al., Human Factors Engineering the TravTek Driver Interface, Vehicular Navigation and Information Systems Conference Proceedings, Part II, Oct. 1991.

Michael Muffat et al., European Cooperation on Dual Mode Route Guidance Perspectives for Advanced Research Partners, Vehicular Navigation and Information Systems Conference Proceedings, Part II, Oct. 1991.

High-Performance Wireless Access Point for the Enterprise, ORINOCO™ AP-100 Access Point for the Enterprise, Lucent Technologies, 2000.

MobileStar Network, MobileStar Network First to Provide Business Travelers with High-Speed Data Access via the Internet—Wirelessly, New York, NY, Jun. 24, 1998.

Harry Chen, et al., "Dynamic Service Discovery for Mobile Computing: Intelligent Agents Meet Jini in the Aether," Cluster Computing, Special Issue on Internet Scalability, vol. 4, No. 4, Feb. 2001.

3rd Generation Partnership Project: Technical Specification Group Services and System Aspects; Functional Stage 2 Description of Location Services in UMTS (1999).

http://www.openwave.com/us/news_room/press_releases/2001/20020320, "Open Wave Announces Availability to End-to-End Set of Location Services for Wireless Internet".

Tremblay, A., "Wireless products arm road warriors," National Underwriter, vol. 105, No. 3, pp. 23-25, Dialog 02113577 67213220 (Jan. 2001).

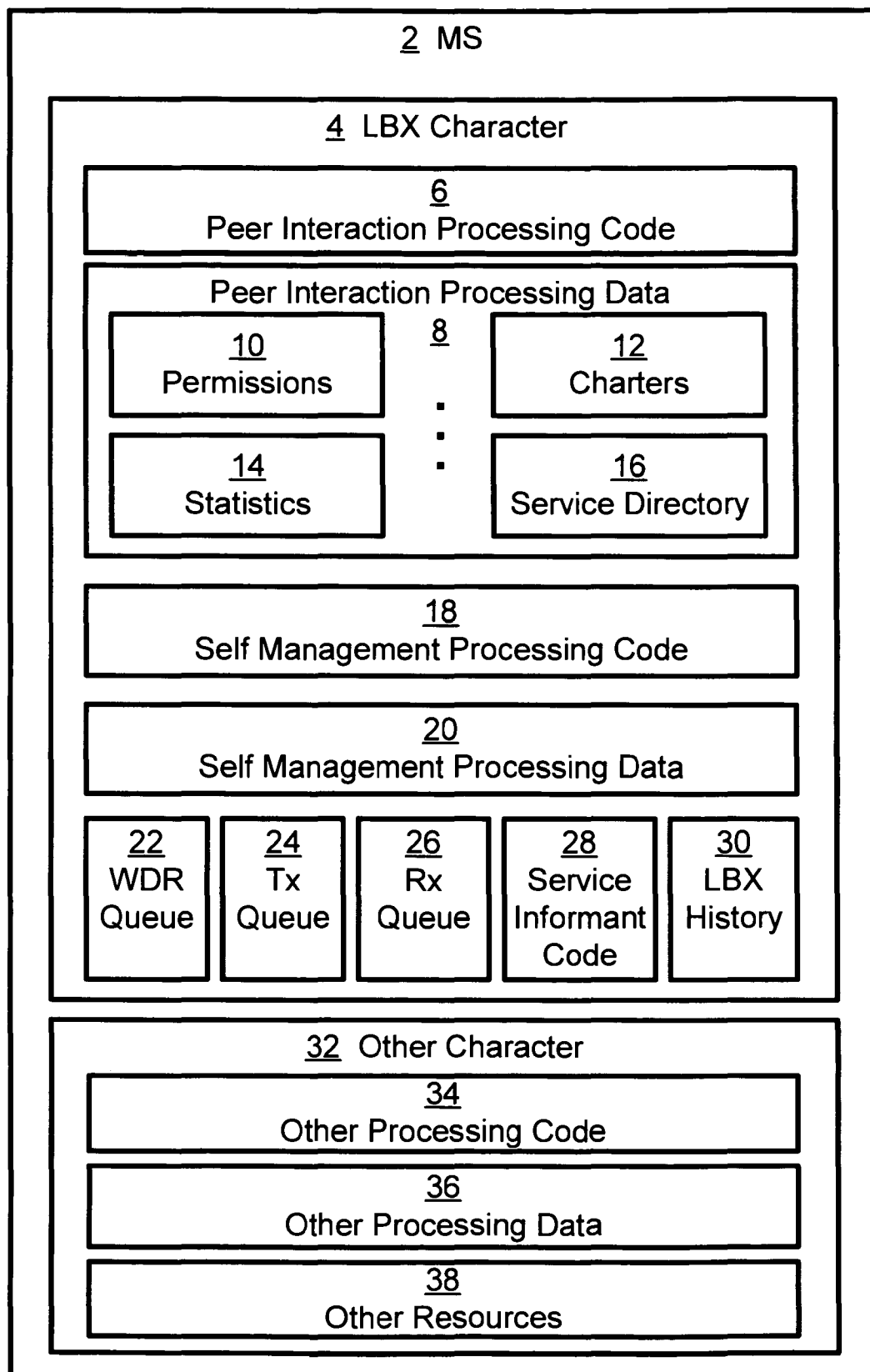
Antonio, Interfaces and Algorithms for a Wide-Area Event Notification Service, Oct. 1999.

U.S. Patent

Nov. 19, 2019

Sheet 1 of 322

US 10,477,994 B2

**Fig. 1A**

U.S. Patent

Nov. 19, 2019

Sheet 2 of 322

US 10,477,994 B2

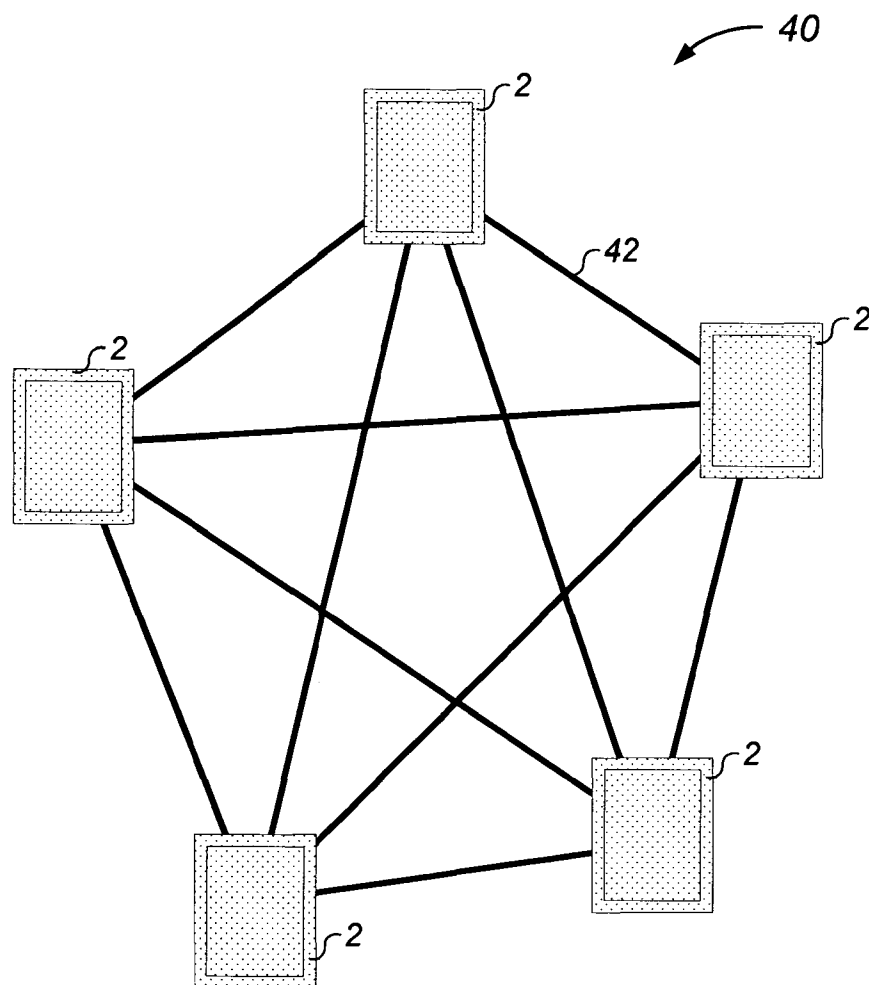


Fig. 1B

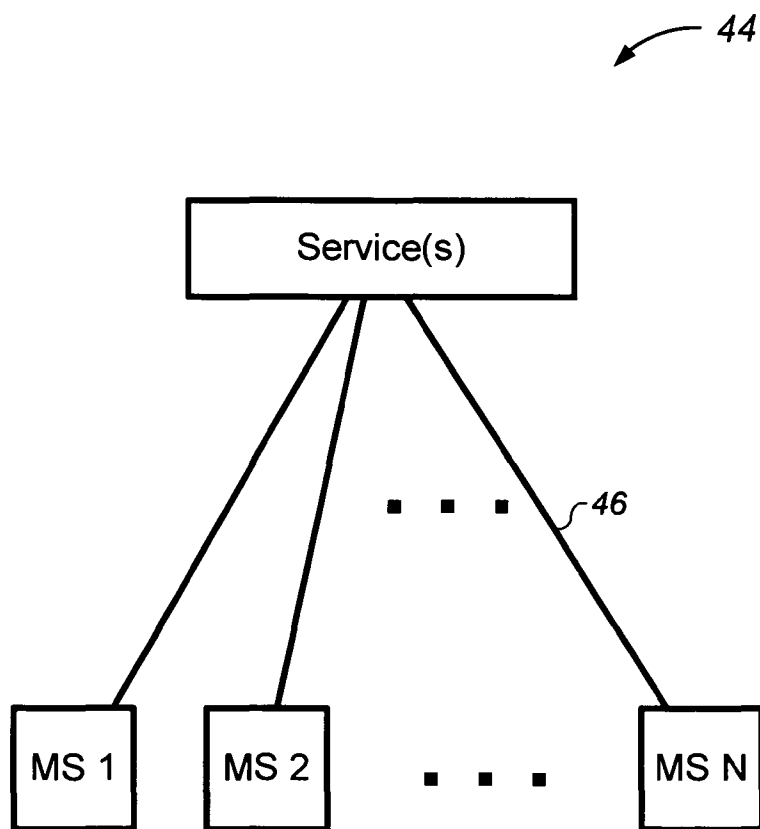


Fig. 1C

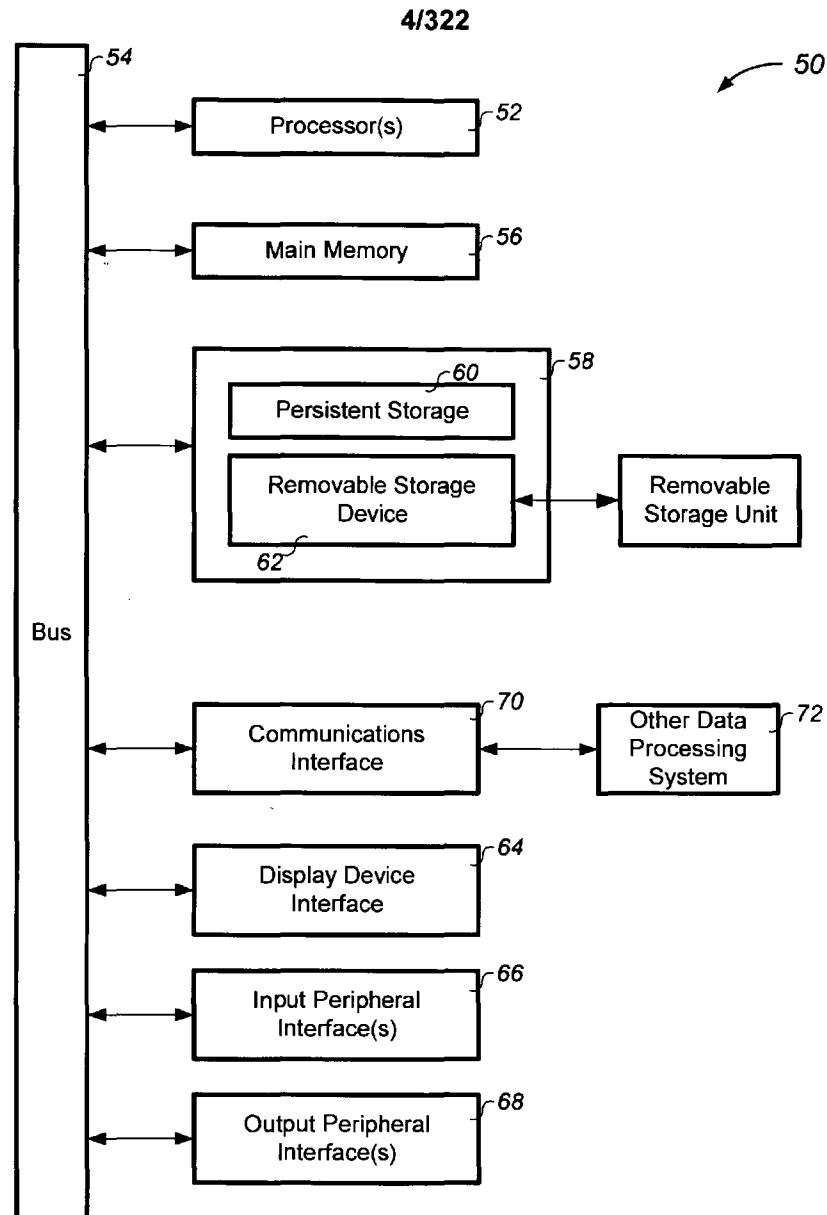
U.S. Patent

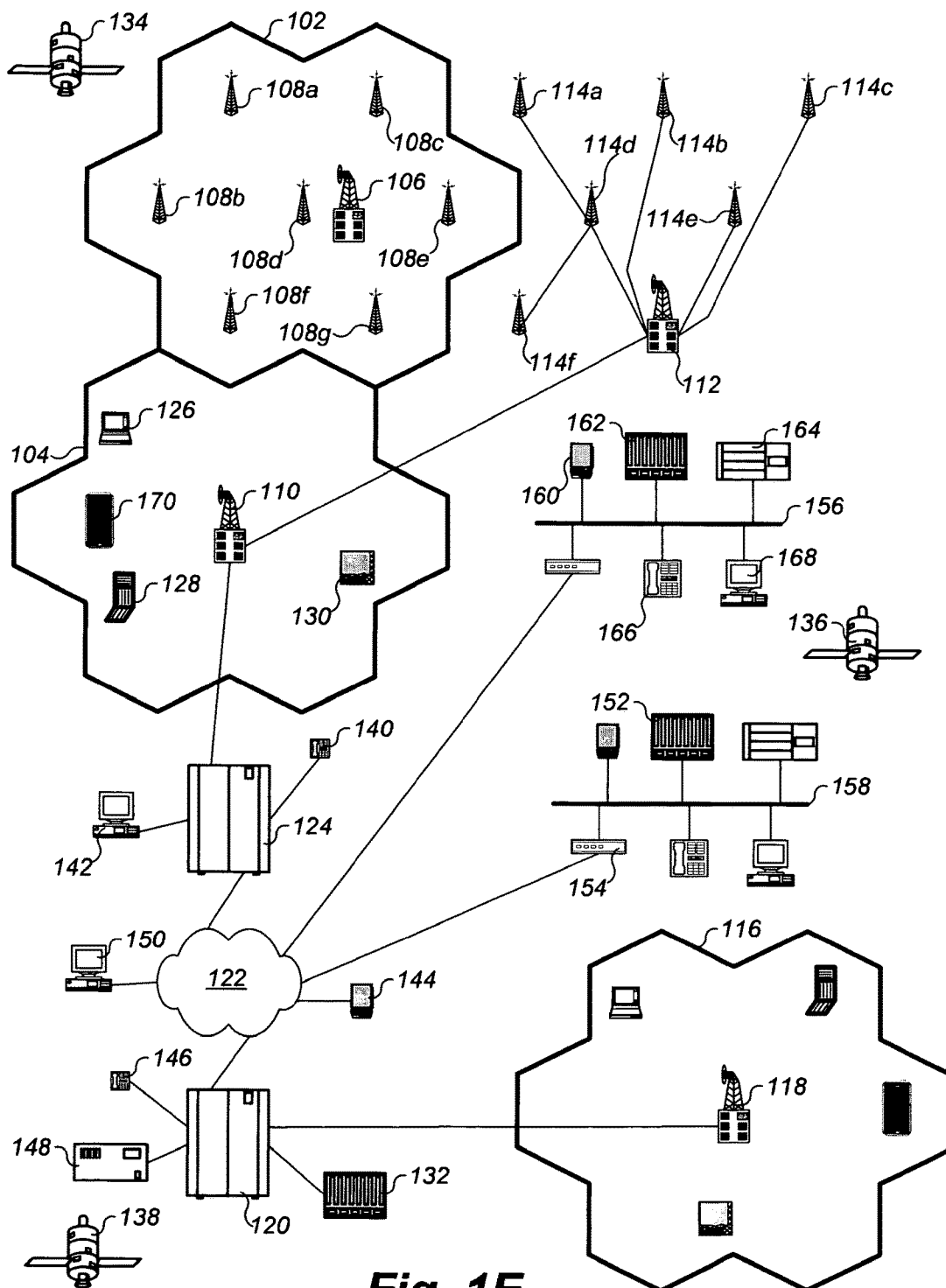
Nov. 19, 2019

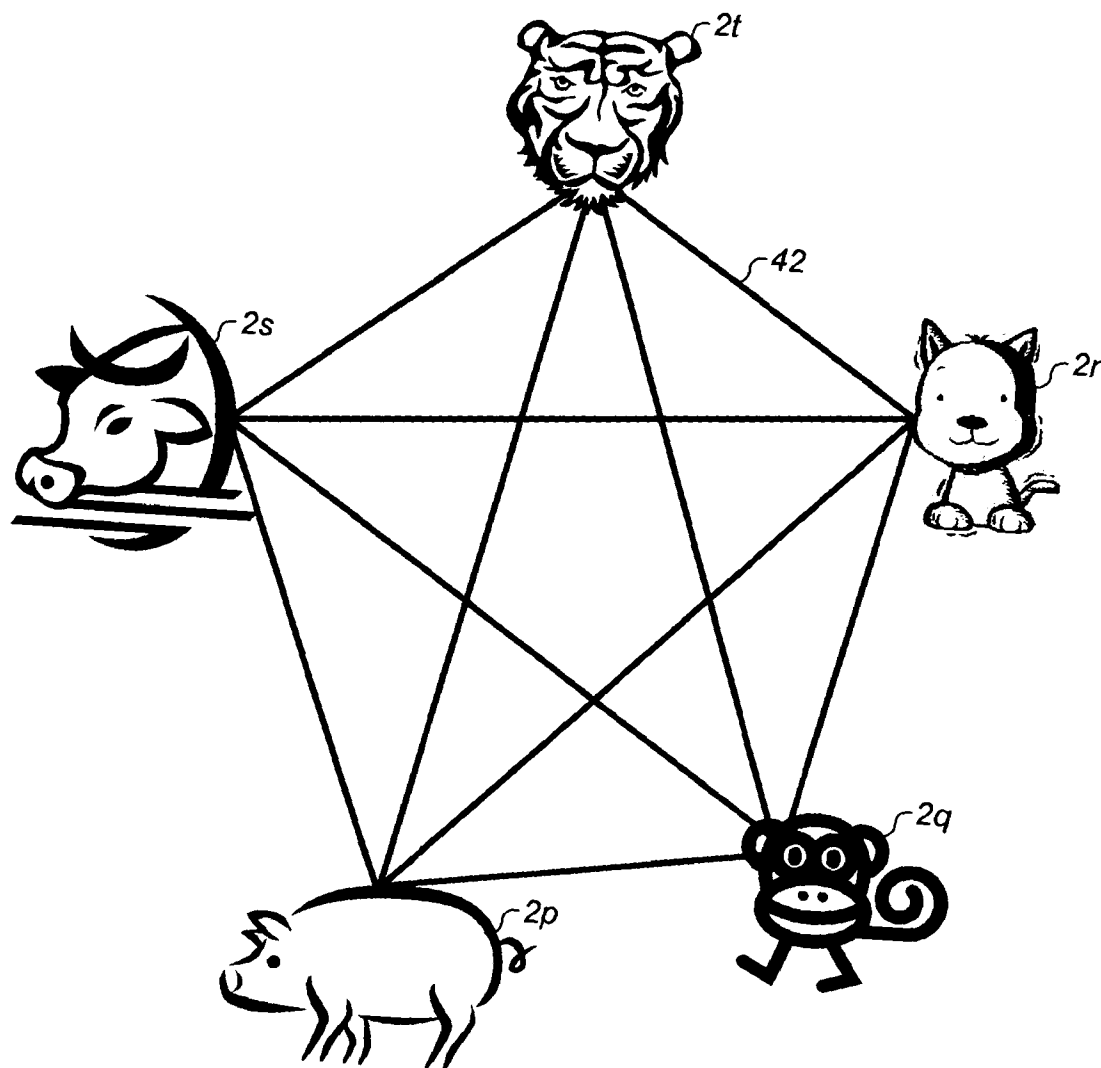
Sheet 4 of 322

US 10,477,994 B2

William J. Johnson WJJ0802

**Fig. 1D**

**Fig. 1E**

**Fig. 1F**

U.S. Patent

Nov. 19, 2019

Sheet 7 of 322

US 10,477,994 B2

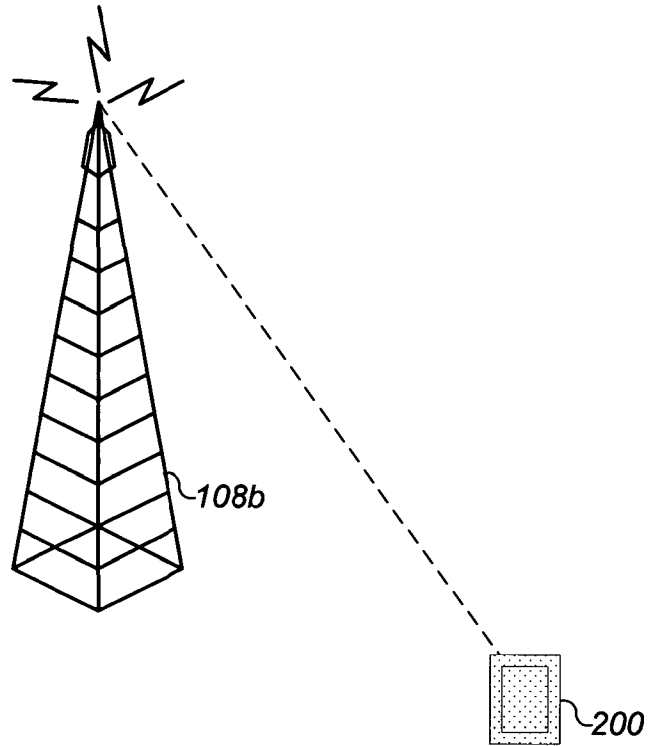


Fig. 2A

U.S. Patent

Nov. 19, 2019

Sheet 8 of 322

US 10,477,994 B2

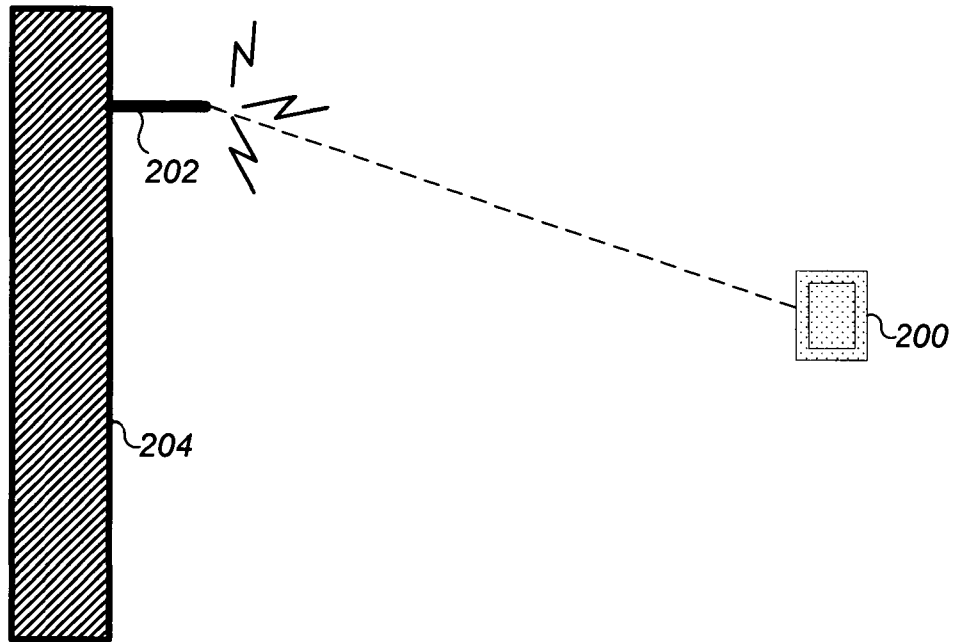


Fig. 2B

U.S. Patent

Nov. 19, 2019

Sheet 9 of 322

US 10,477,994 B2

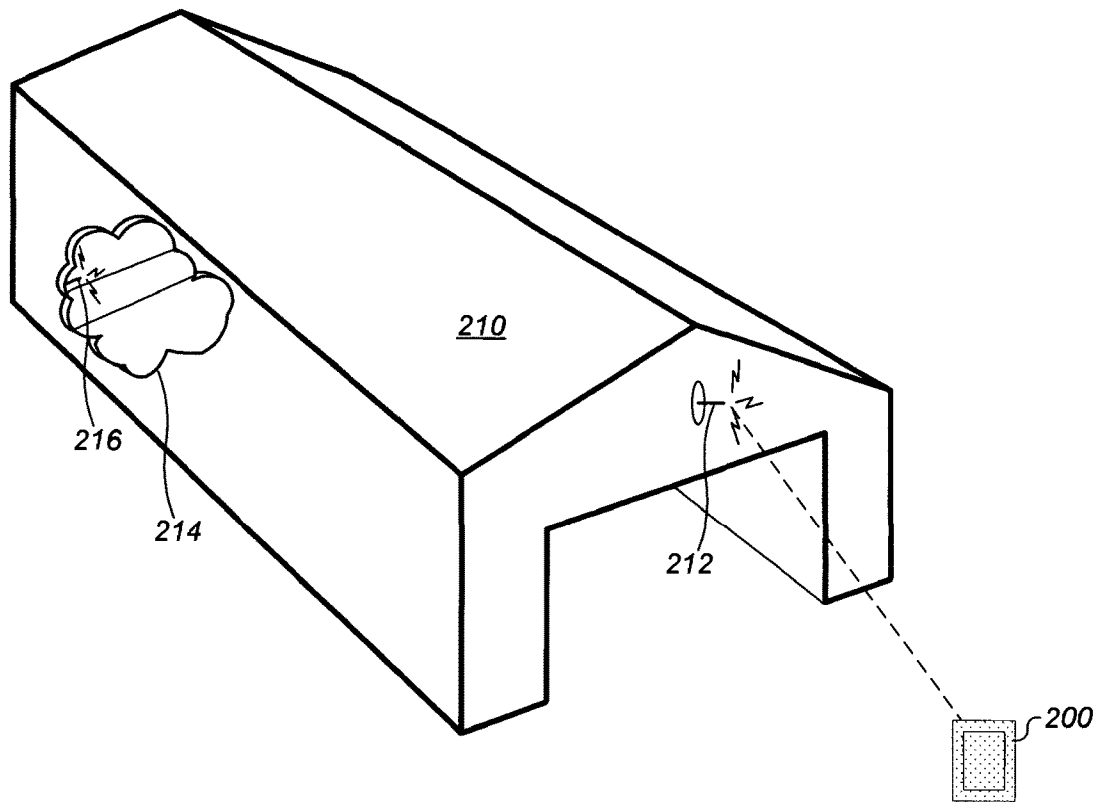
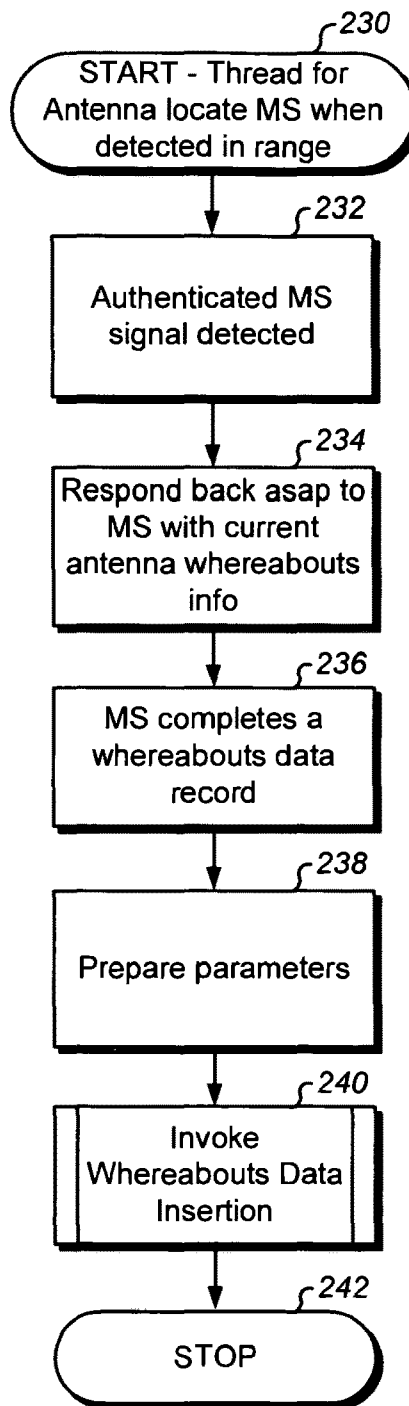
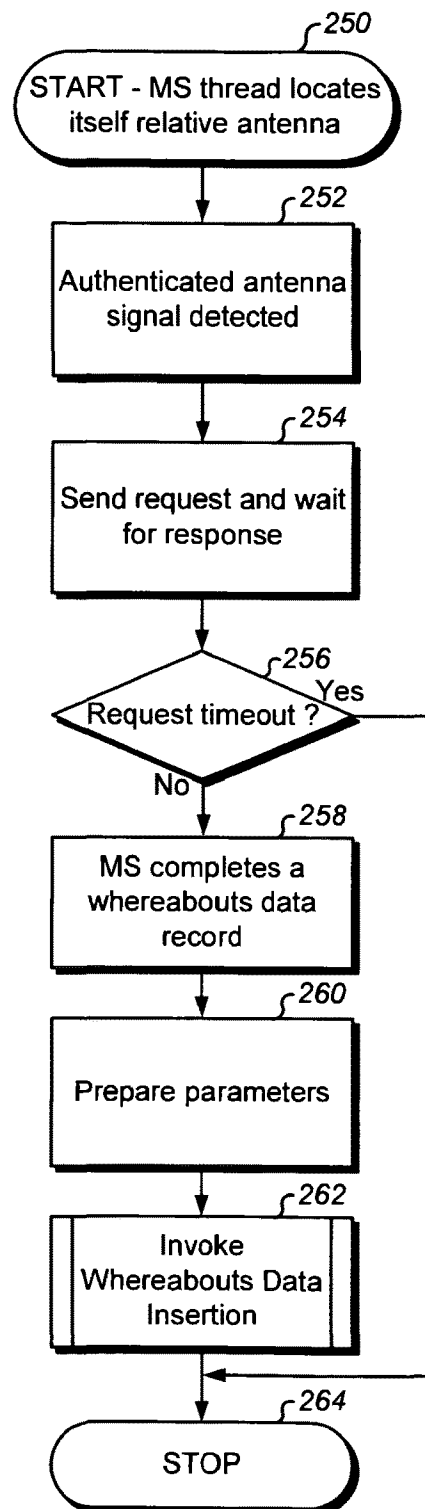
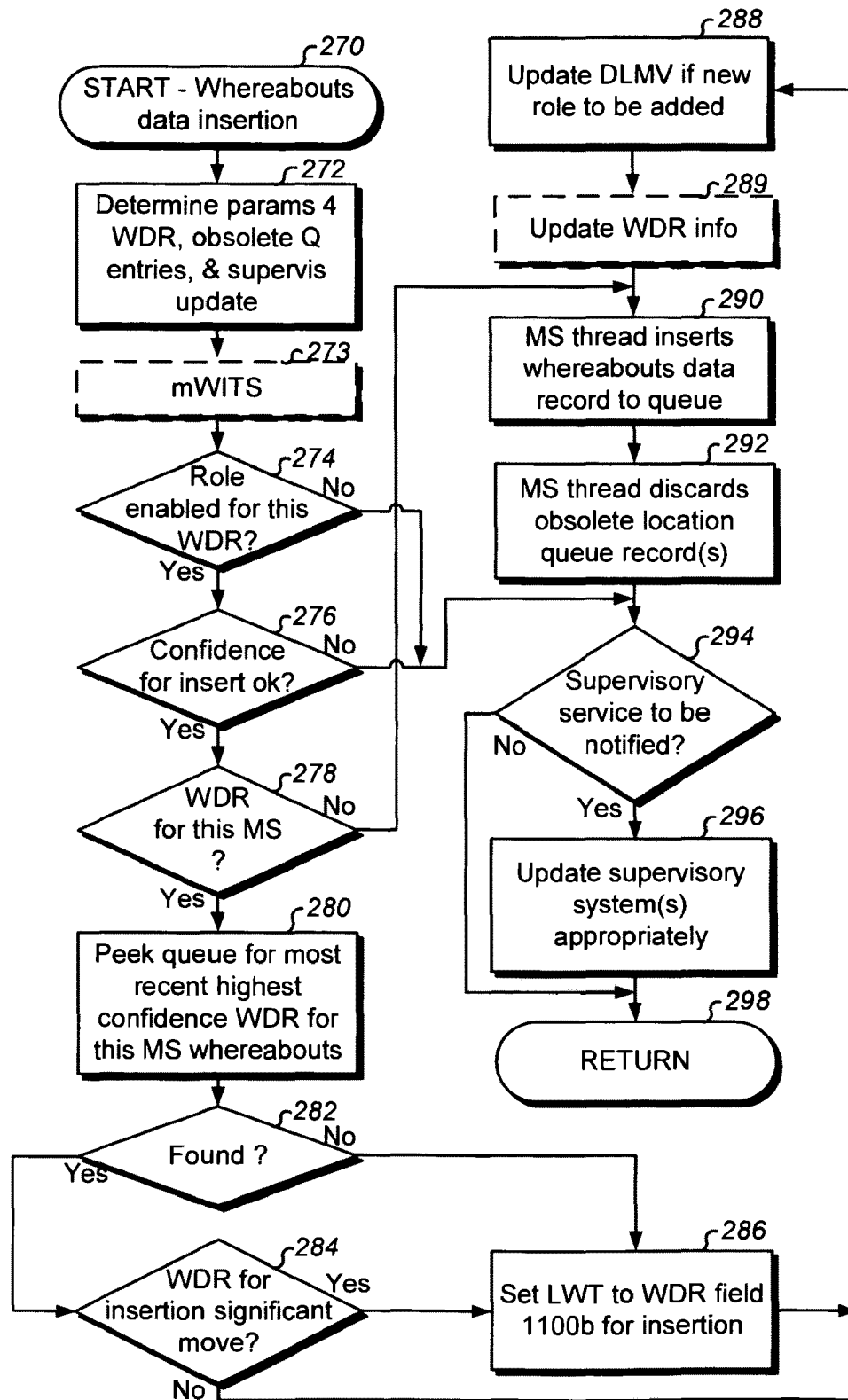


Fig. 2C

**Fig. 2D**

**Fig. 2E**

**Fig. 2F**

U.S. Patent

Nov. 19, 2019

Sheet 13 of 322

US 10,477,994 B2

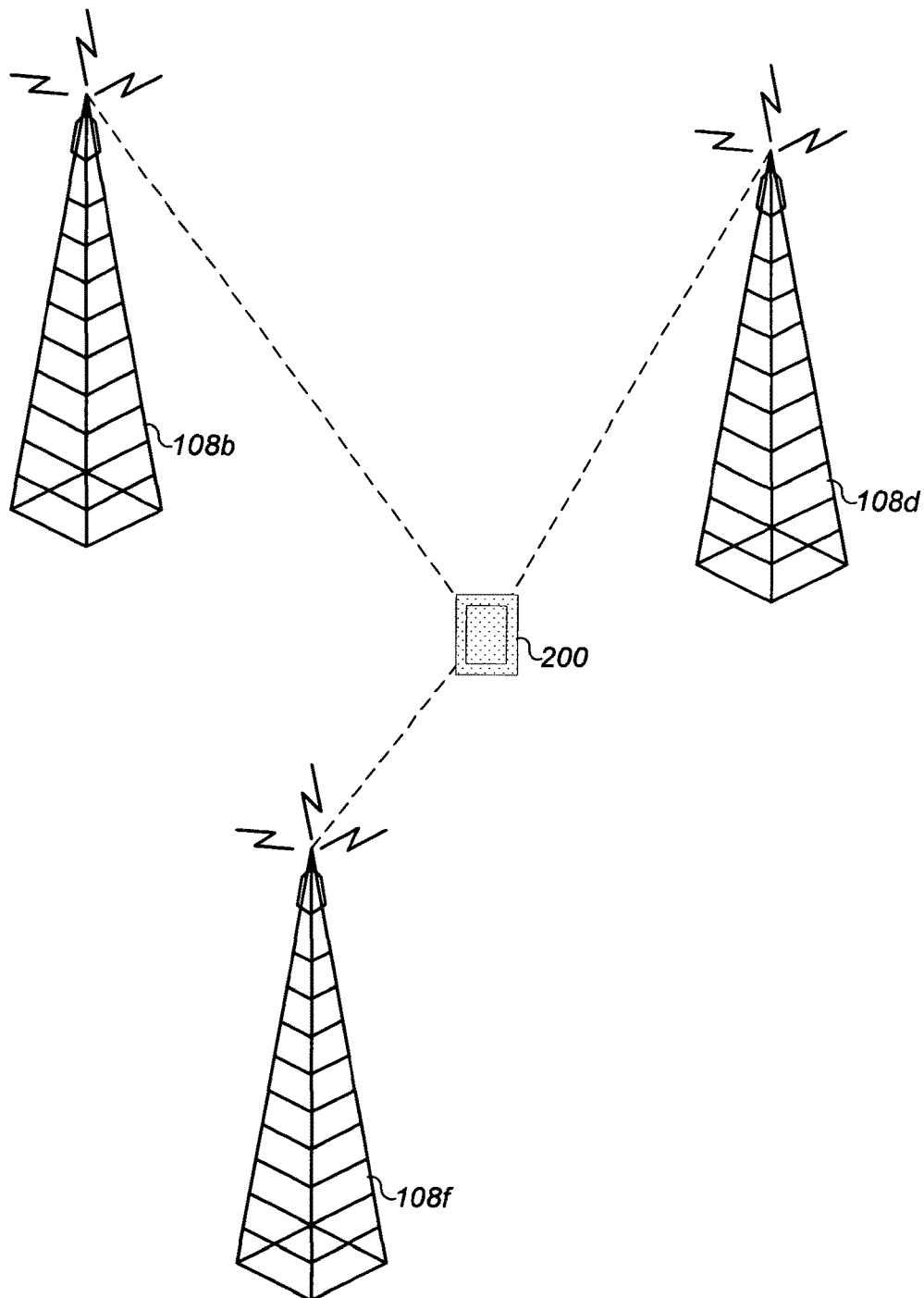
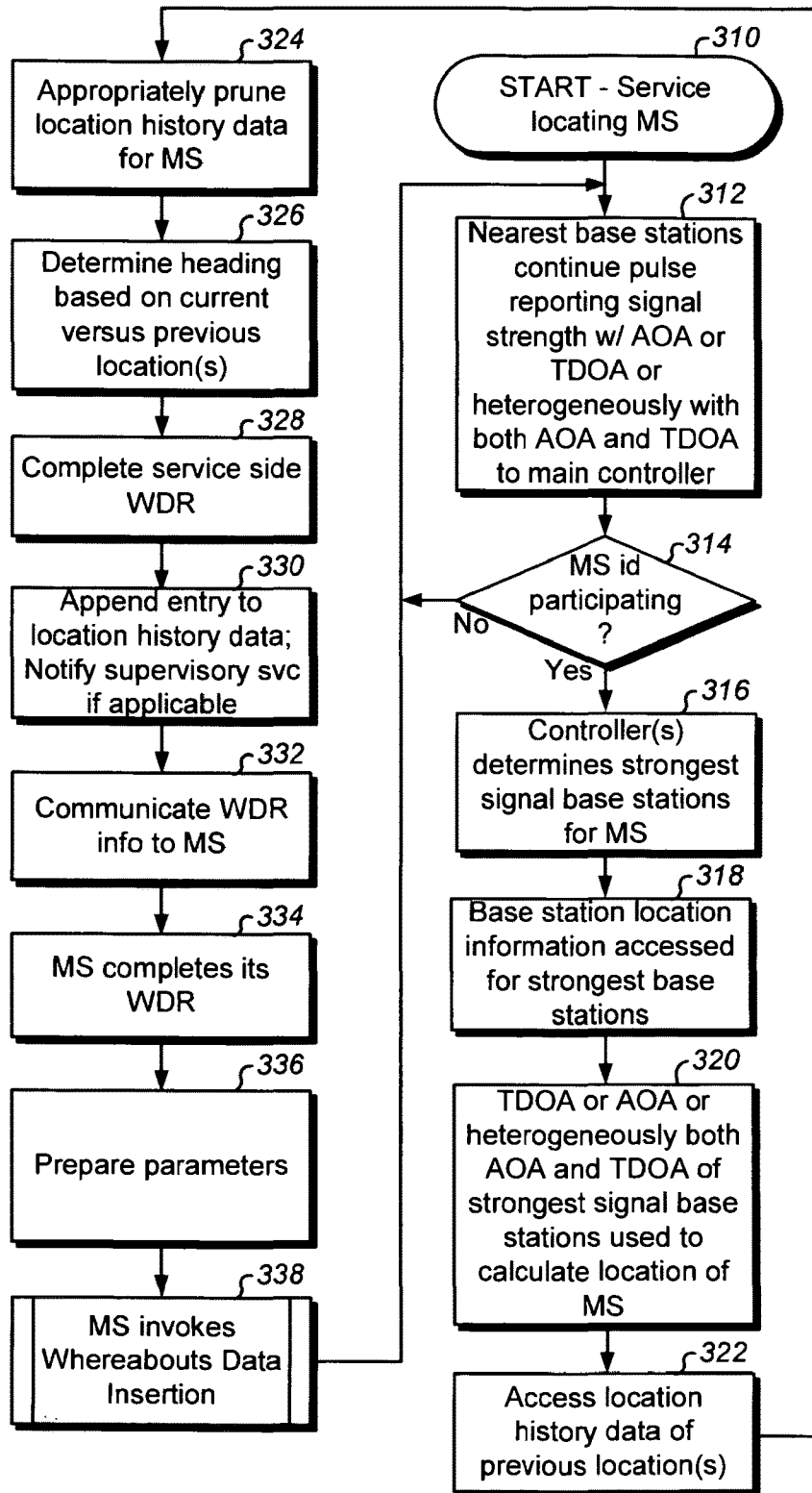
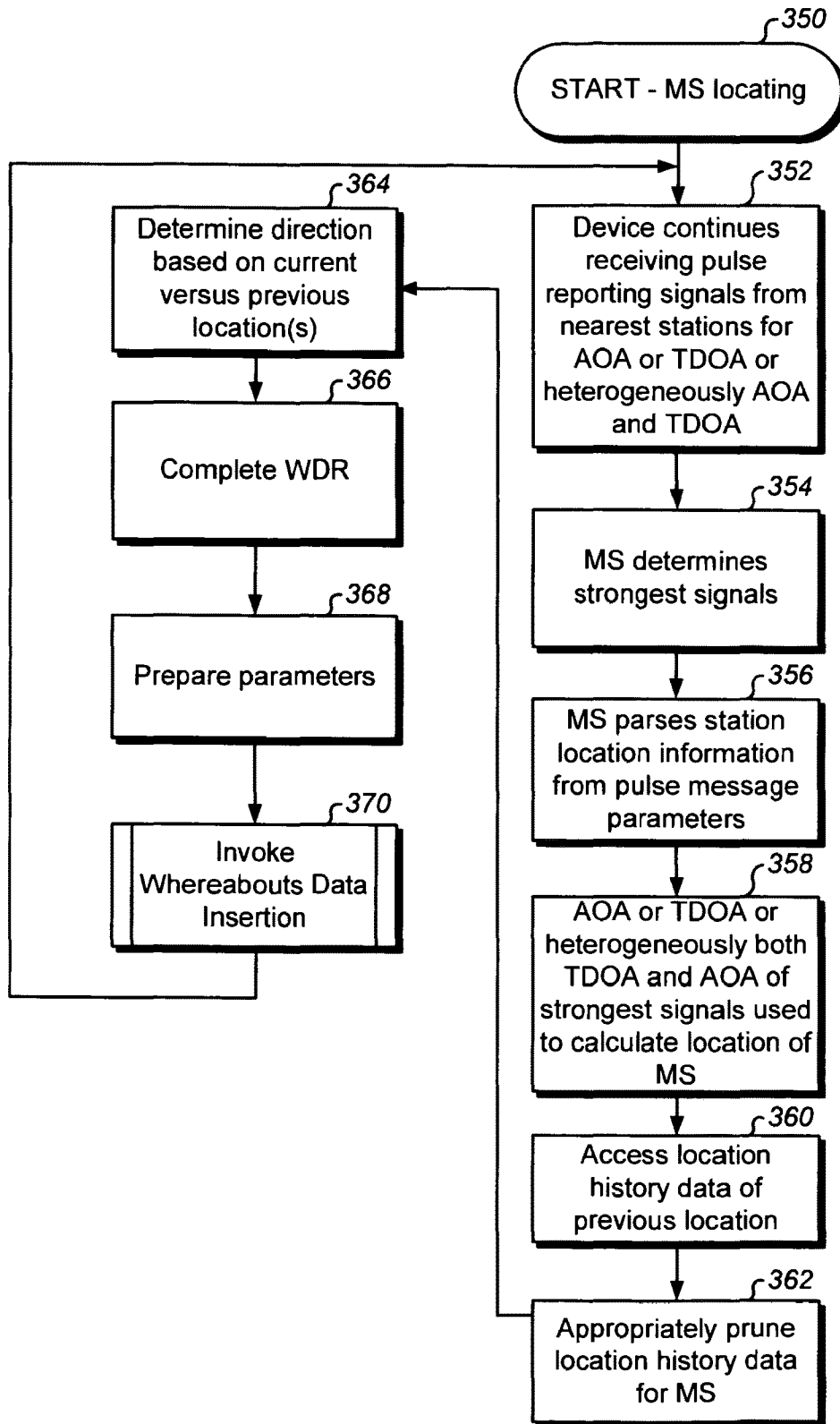


Fig. 3A

**Fig. 3B**

**Fig. 3C**

U.S. Patent

Nov. 19, 2019

Sheet 16 of 322

US 10,477,994 B2

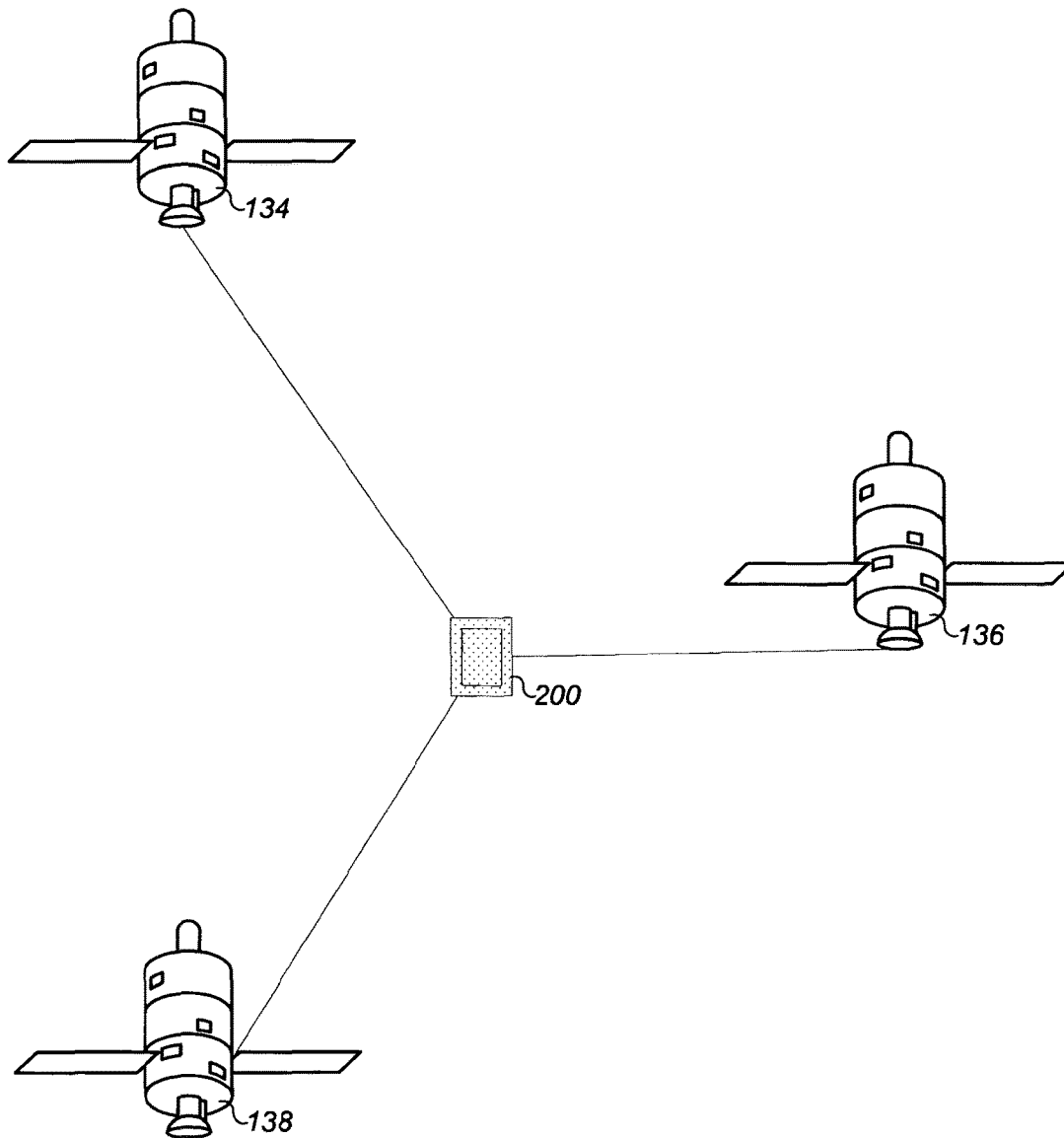
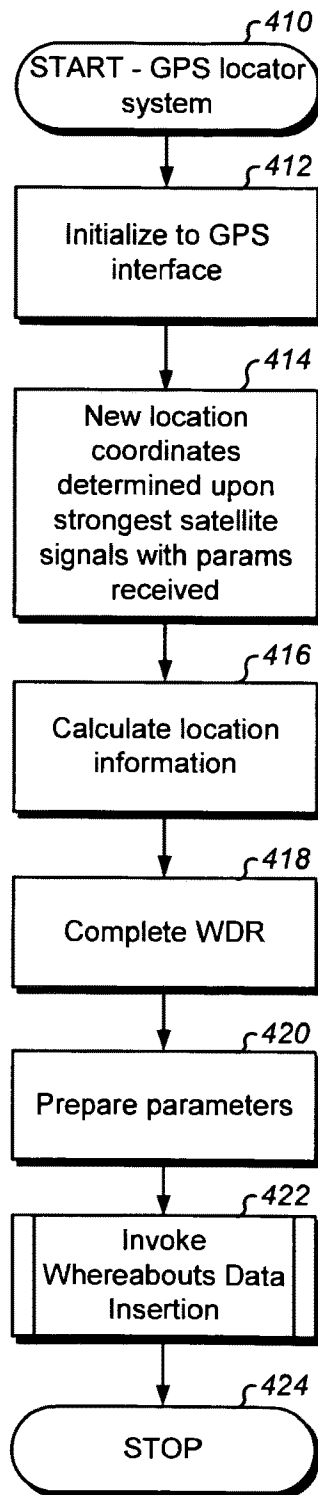


Fig. 4A

**Fig. 4B**

U.S. Patent

Nov. 19, 2019

Sheet 18 of 322

US 10,477,994 B2

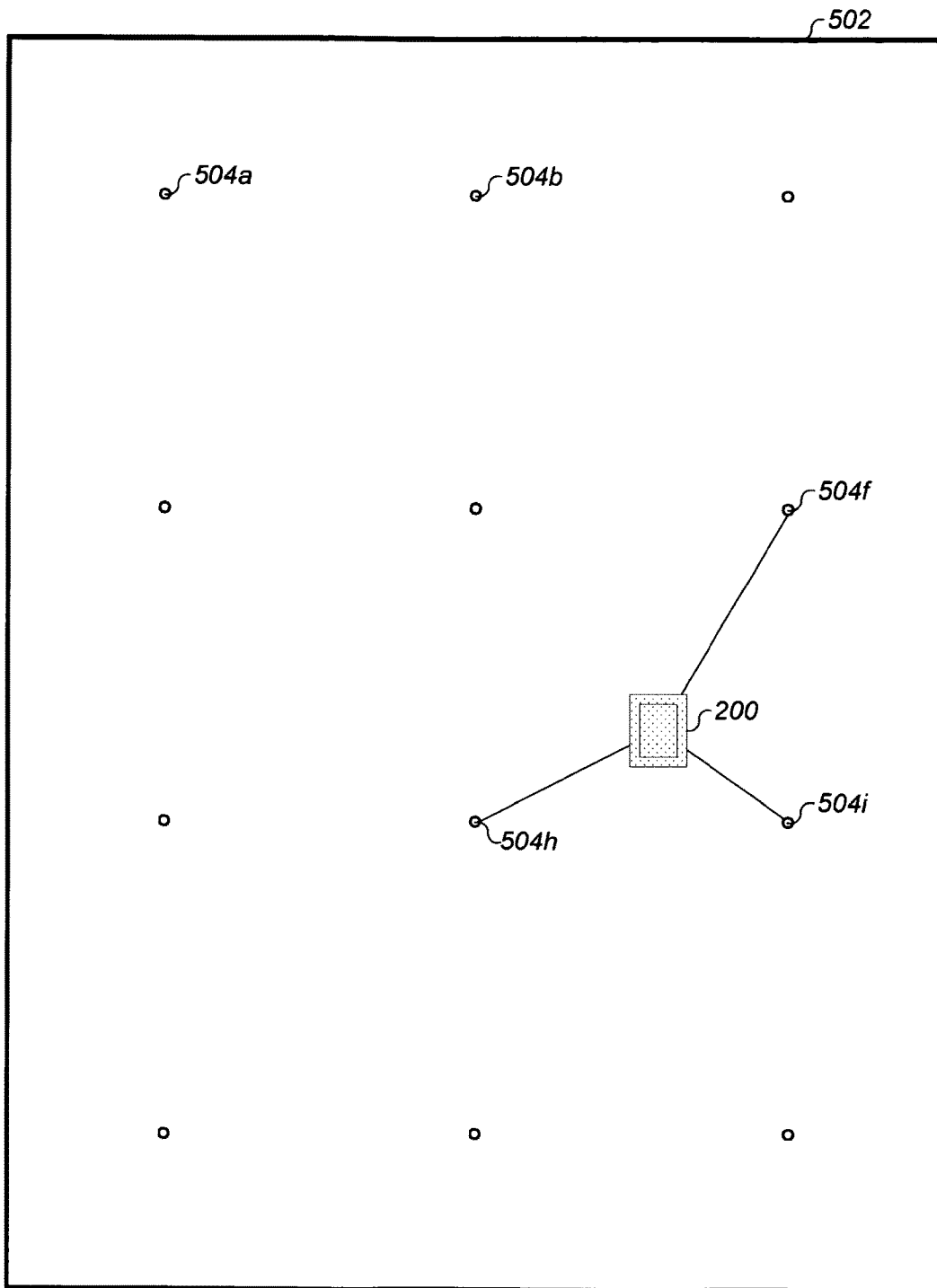
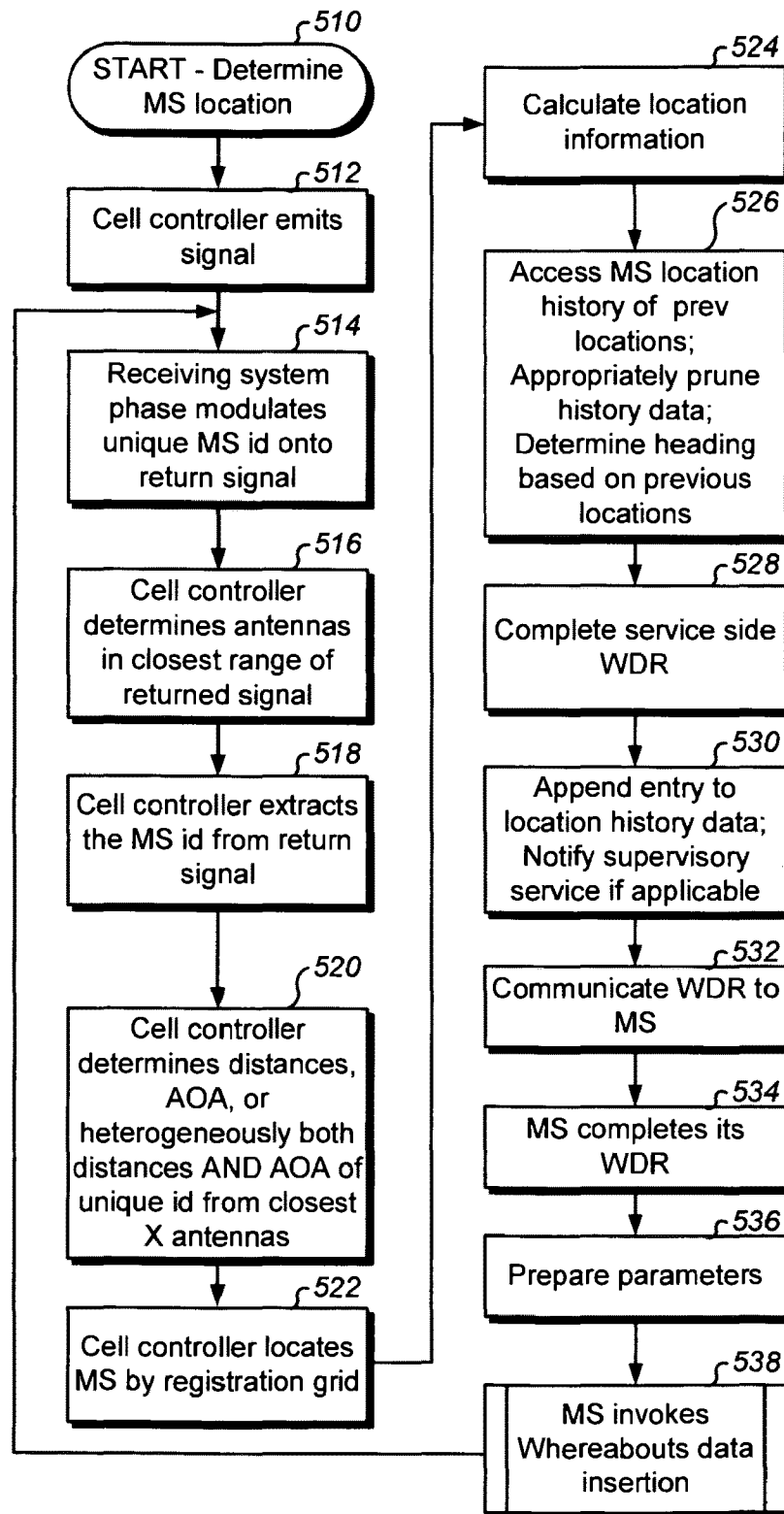


Fig. 5A

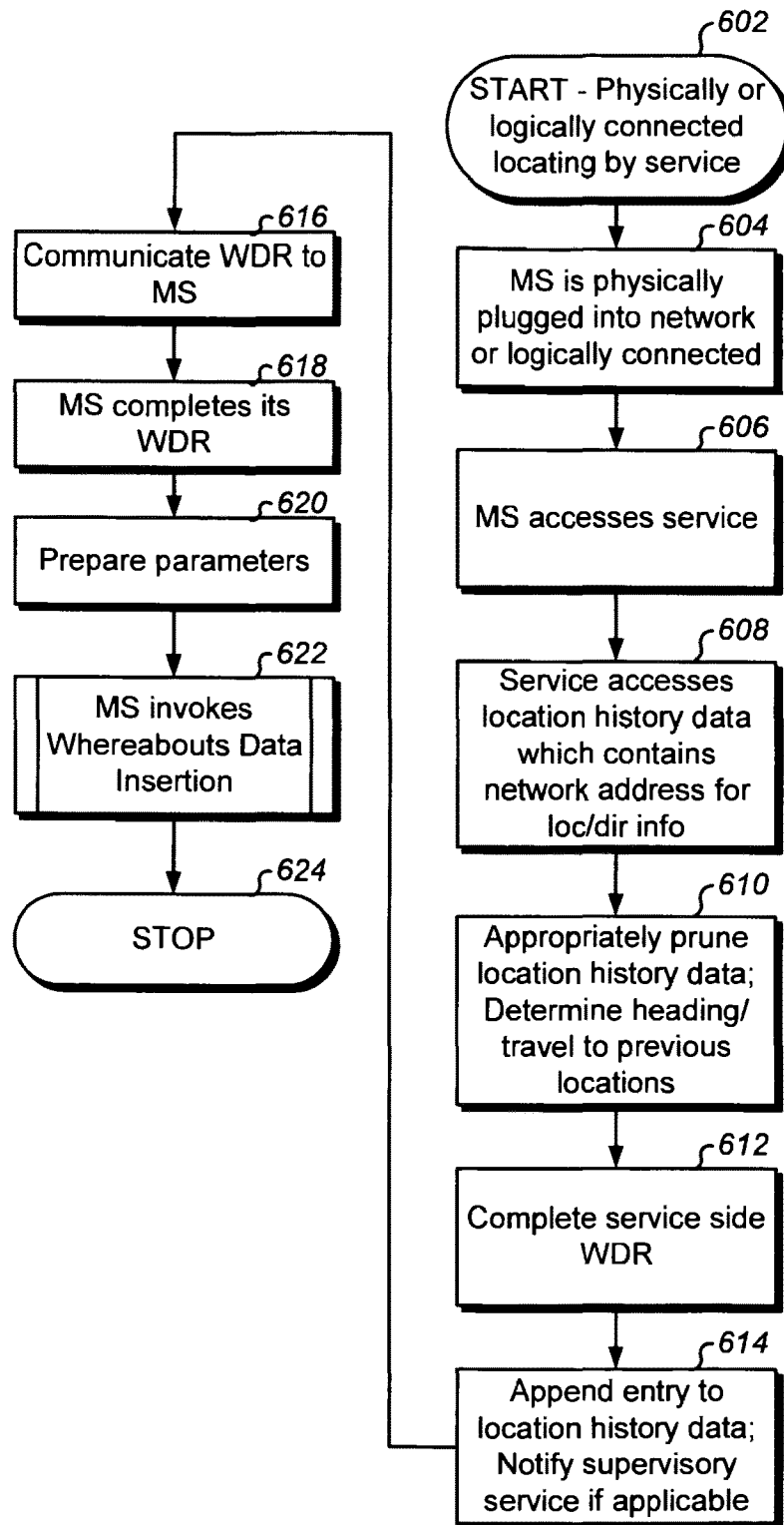
**Fig. 5B**

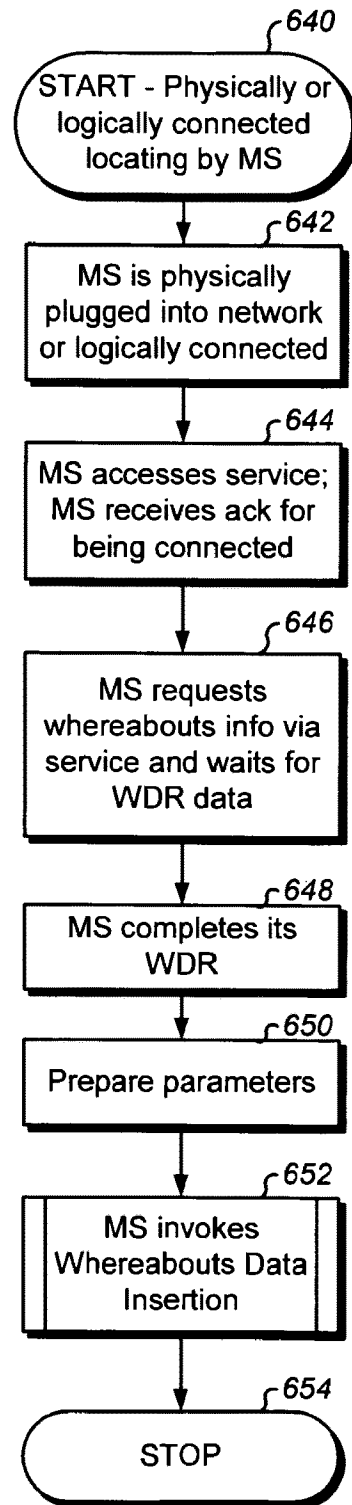
U.S. Patent

Nov. 19, 2019

Sheet 20 of 322

US 10,477,994 B2

**Fig. 6A**

**Fig. 6B**

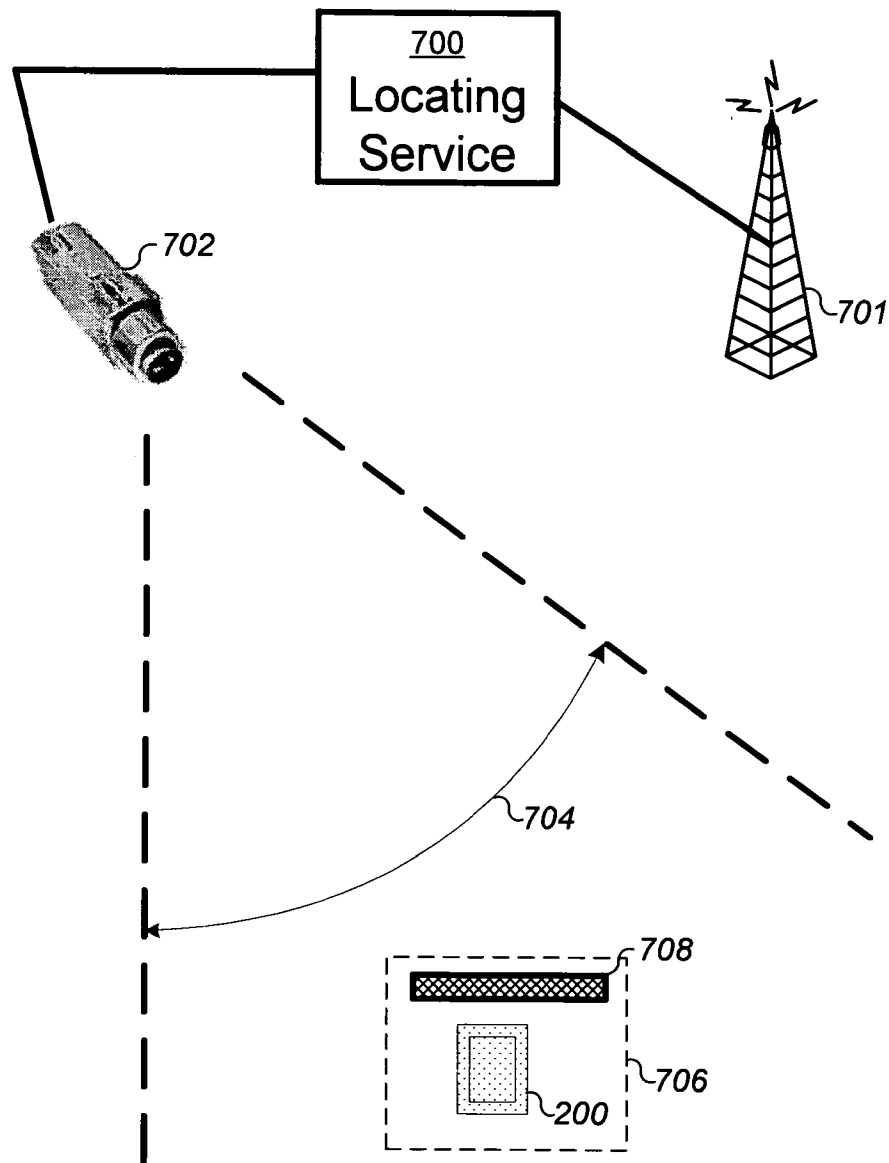


Fig. 7A

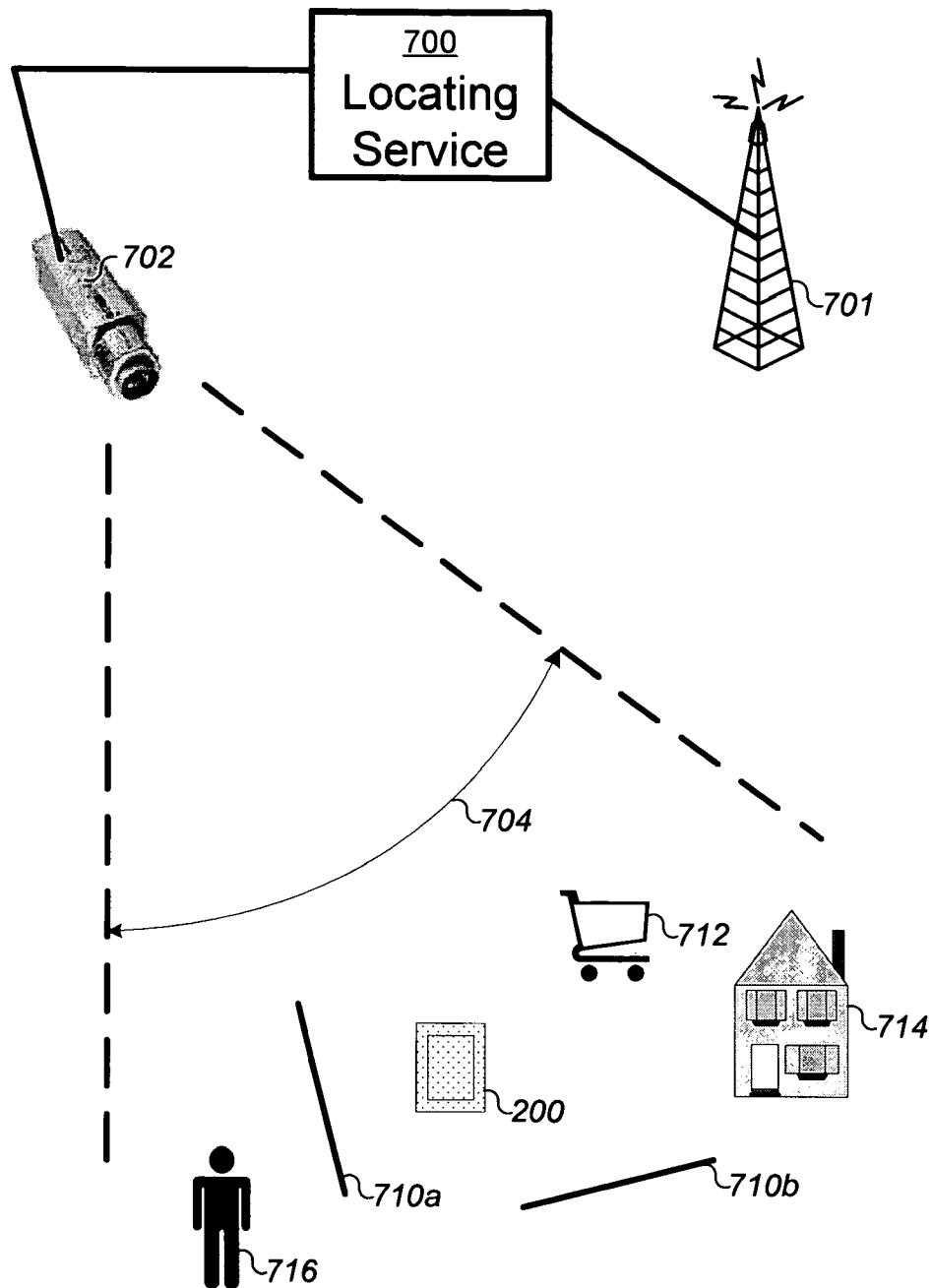


Fig. 7B

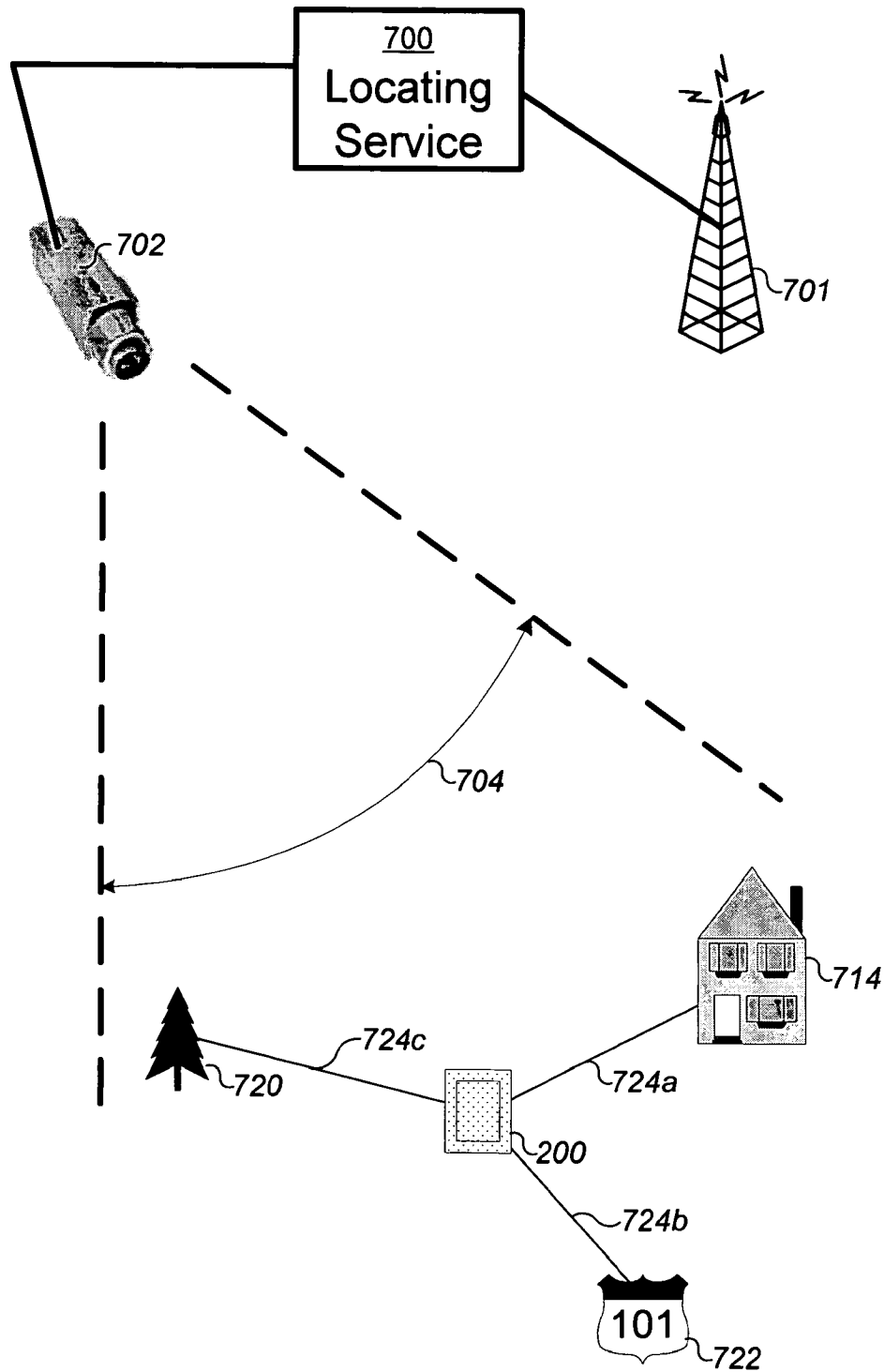
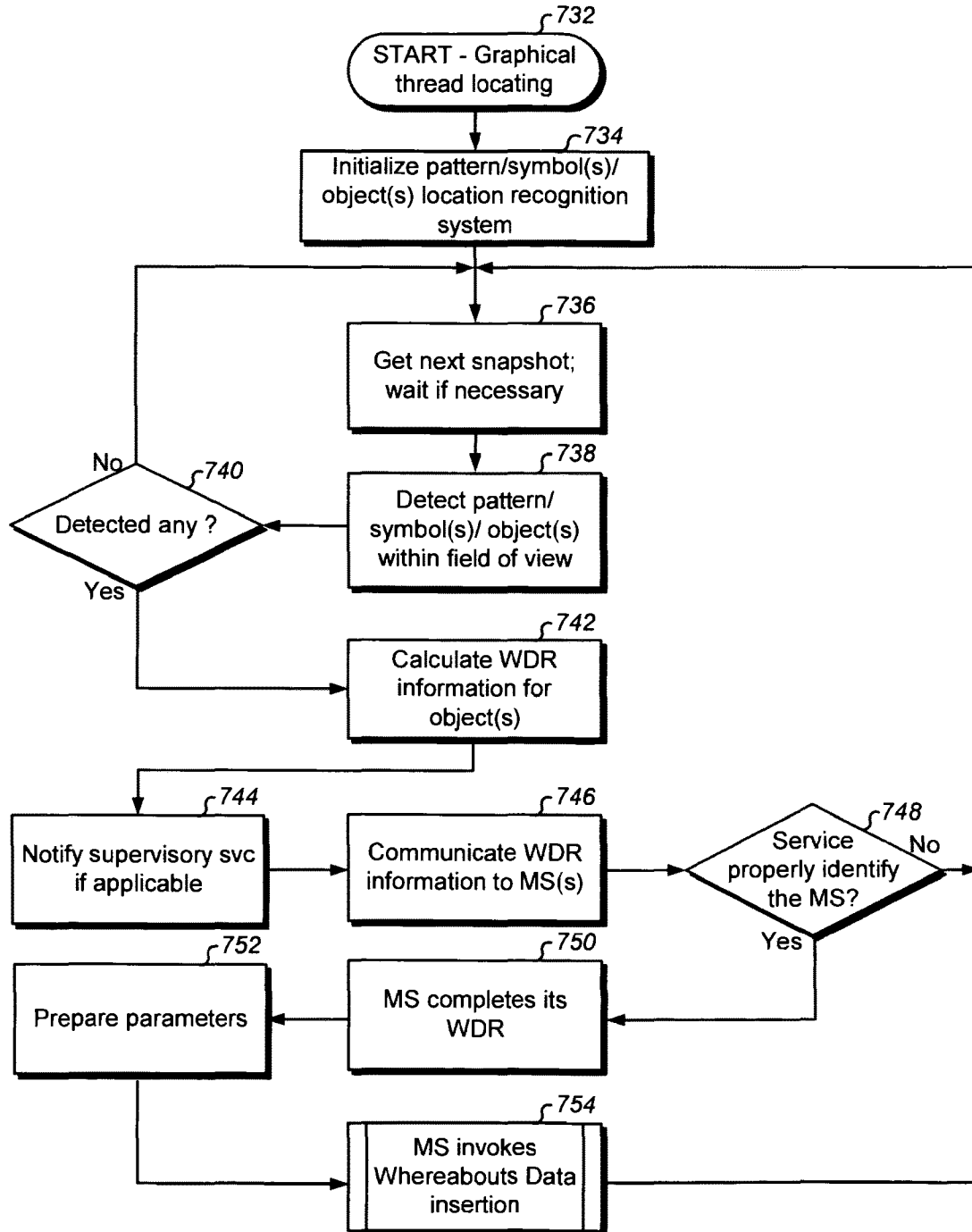


Fig. 7C

**Fig. 7D**

U.S. Patent

Nov. 19, 2019

Sheet 26 of 322

US 10,477,994 B2

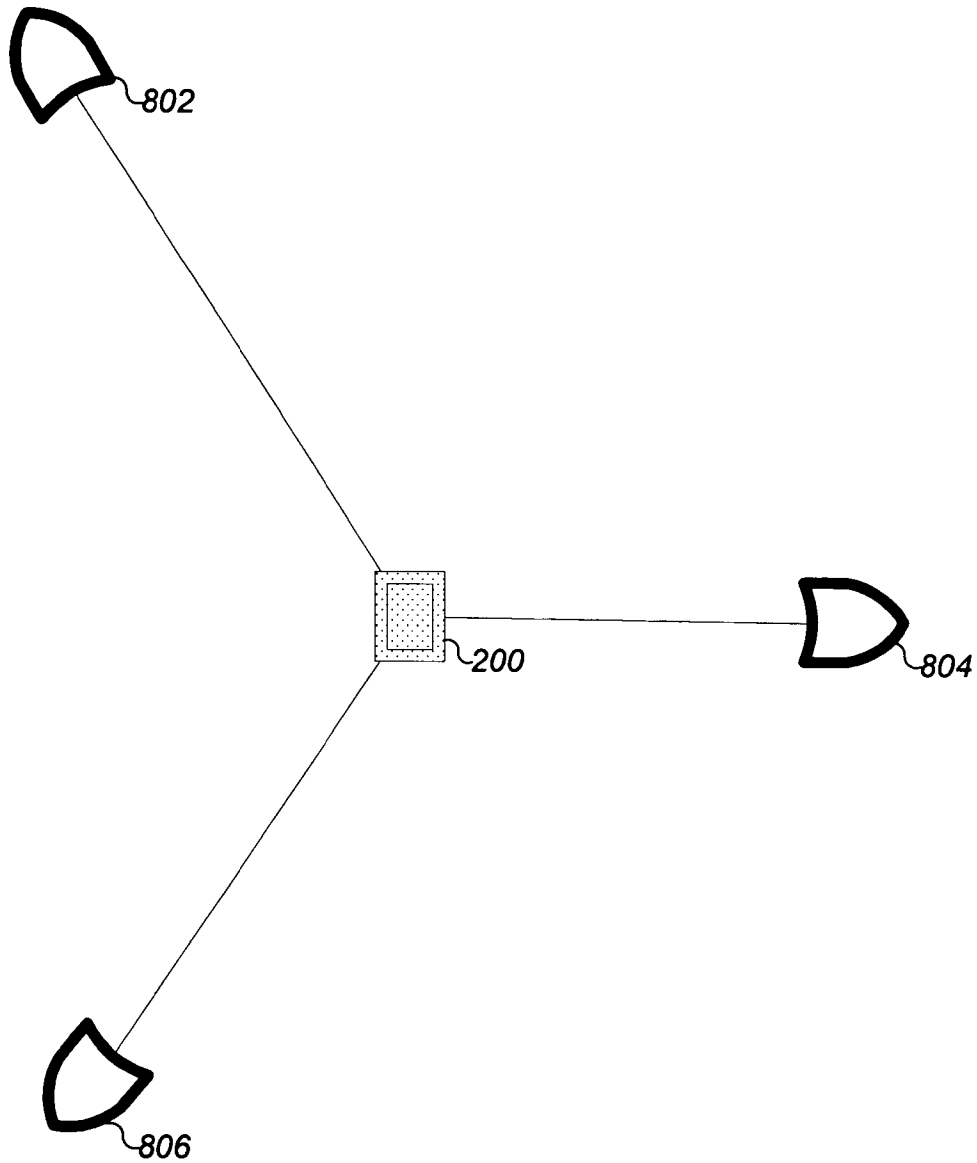
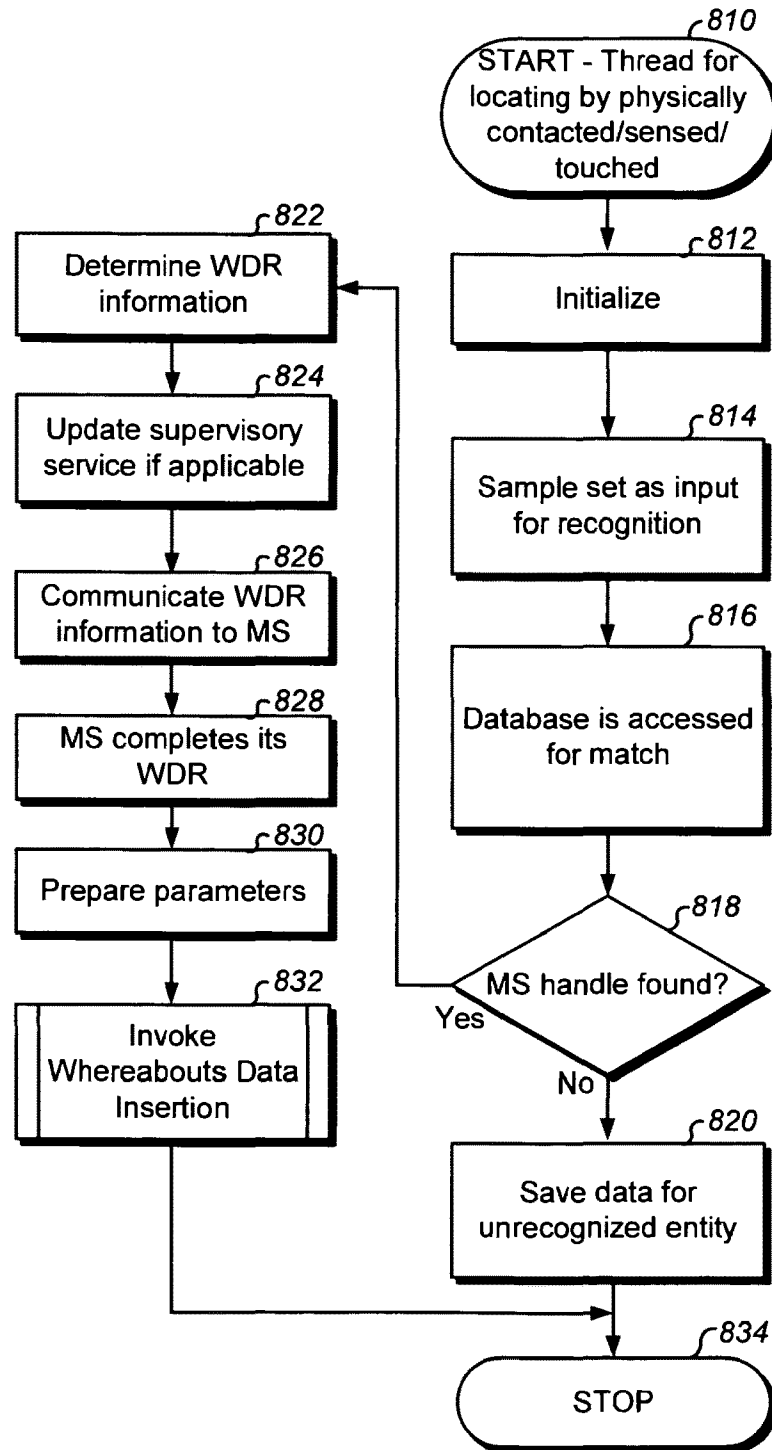
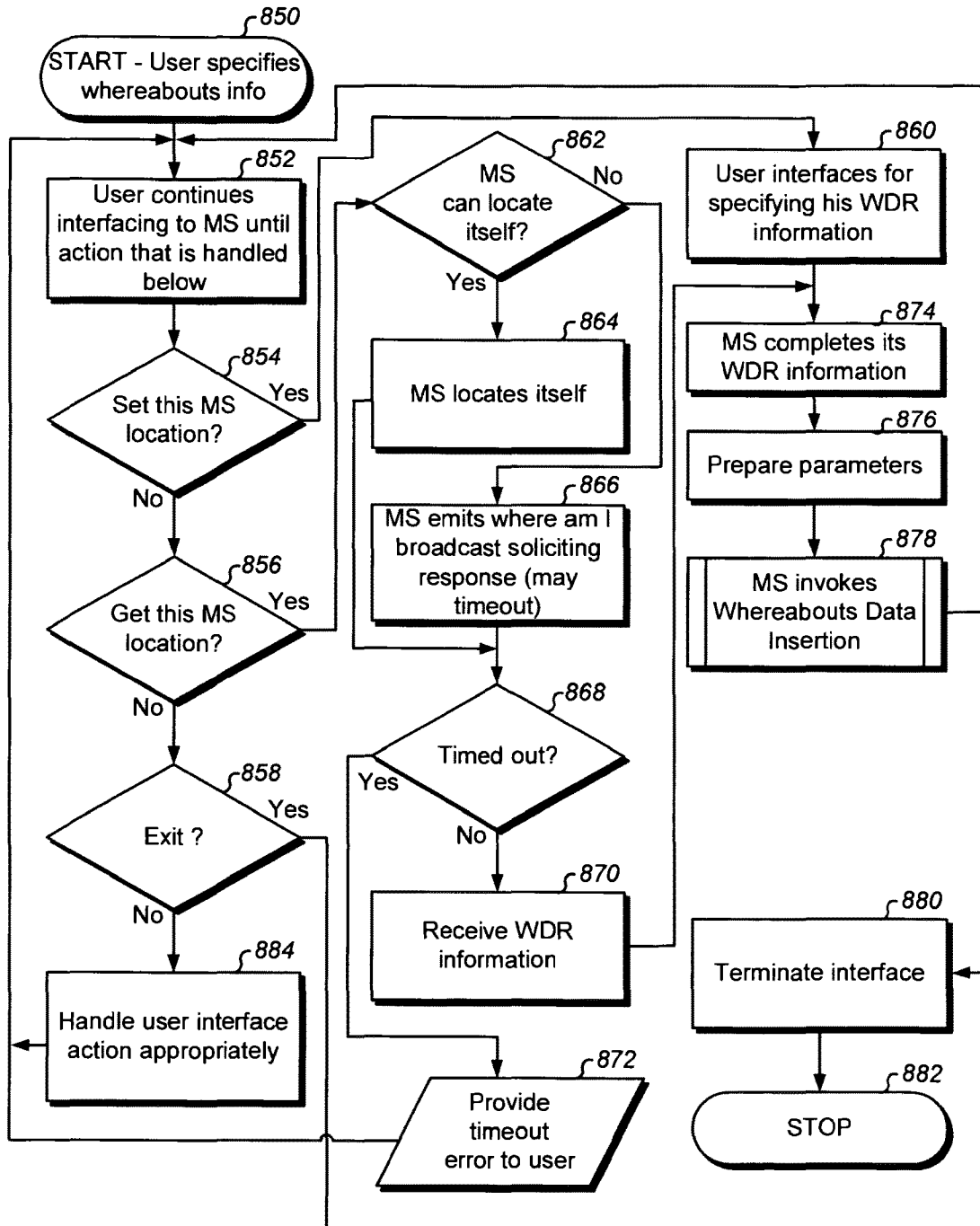


Fig. 8A

**Fig. 8B**

**Fig. 8C**

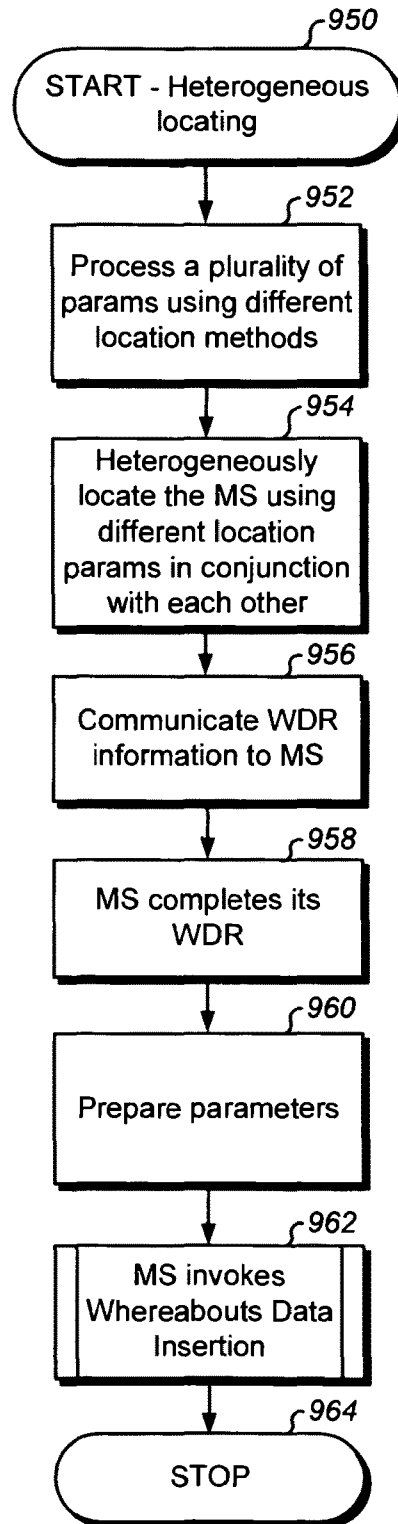
U.S. Patent

Nov. 19, 2019

Sheet 29 of 322

US 10,477,994 B2

		MS (id 0A12:43EF:985B:012F)
GPS	C	x
	S	
A-GPS	C	
	S	
D-GPS	C	
	S	
Graphic-Pattern(s)	C	
	S	
Graphic-Distances	C	
	S	
Graphic-Triangulate	C	
	S	
Artificial Intelligence	C	
	S	
Cell Range	C	
	S	x
Cell AOA	C	
	S	
Cell TDOA	C	
	S	x
Cell MPT	C	
	S	x
Antenna Range	C	
	S	x
Antenna AOA	C	
	S	x
Antenna TDOA	C	
	S	x
Antenna MPT	C	
	S	x
LIDAR/optics	C	
	S	
Manual	C	
	S	
Contact	C	
	S	x
MPT	C	
	S	x
Client Logical Connect	C	
	S	
Server Logical Connect	C	
	S	
Client Physical Connect	C	
	S	
Server Physical Connect	C	
	S	
Sound/Acoustics	C	
	S	
Microdot/ RFI	C	
	S	
Transponder	C	
	S	
Others	C	
	S	
...	C	
	S	
Fig. 9A		

**Fig. 9B**

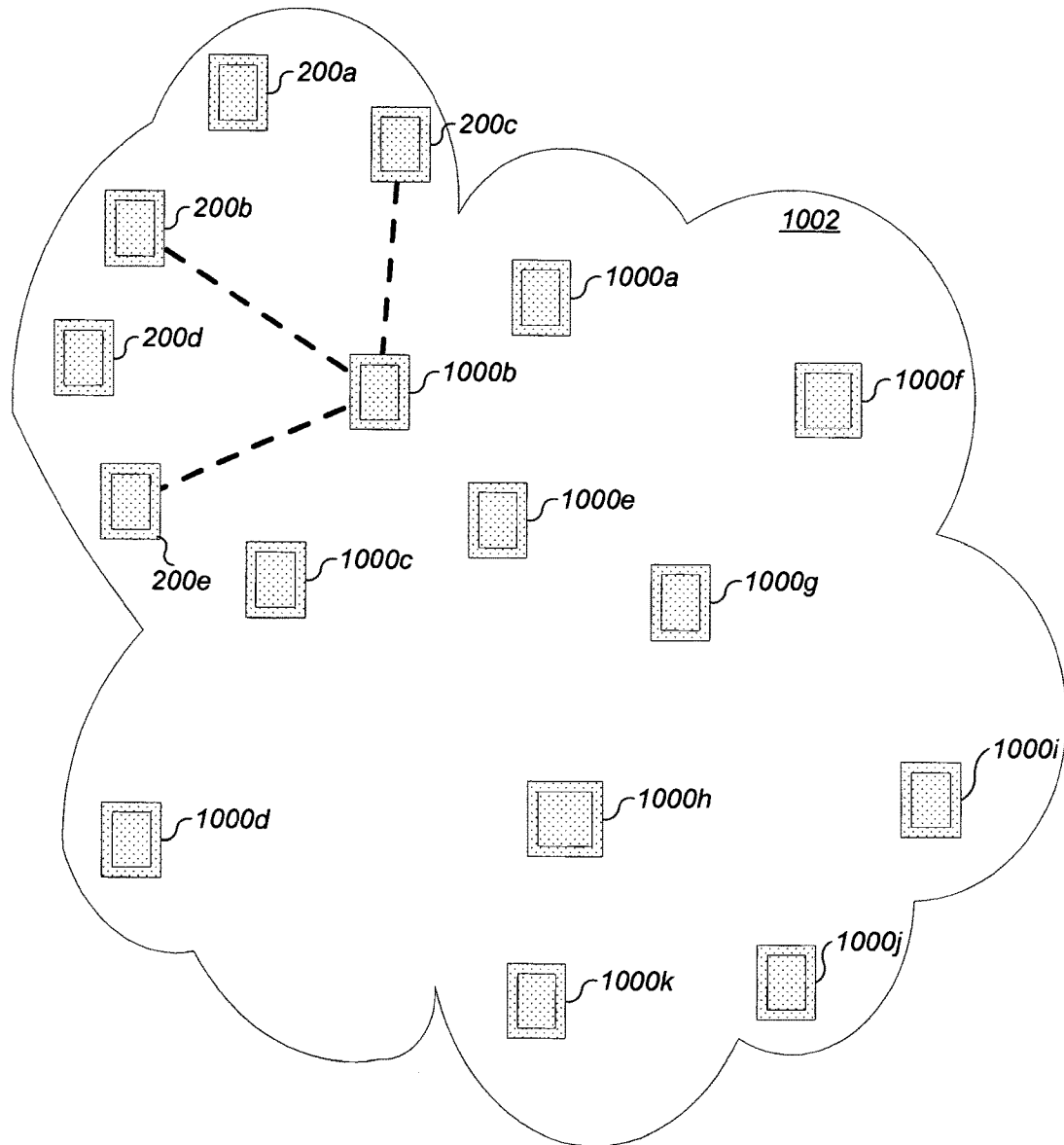


Fig. 10A

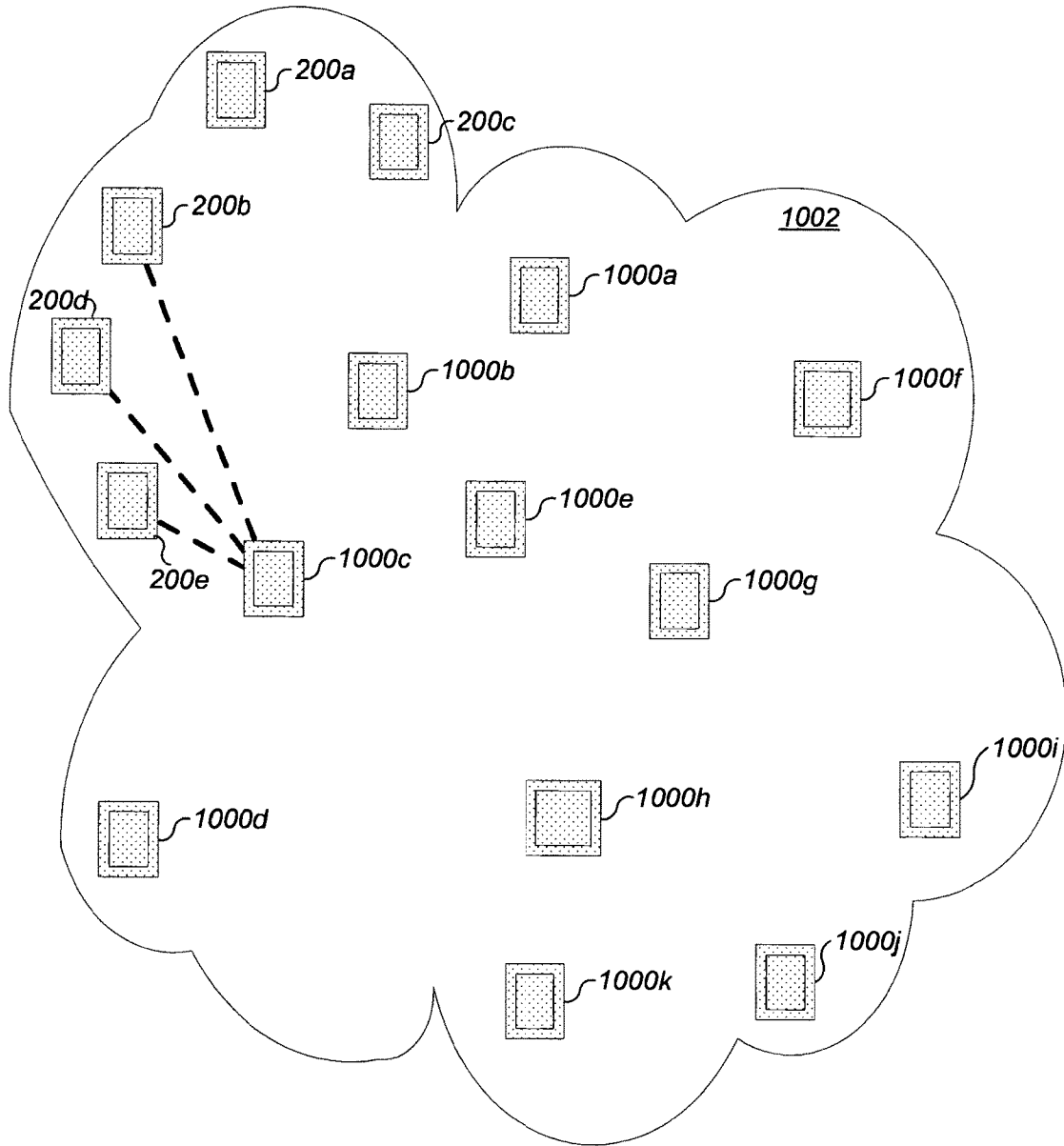


Fig. 10B

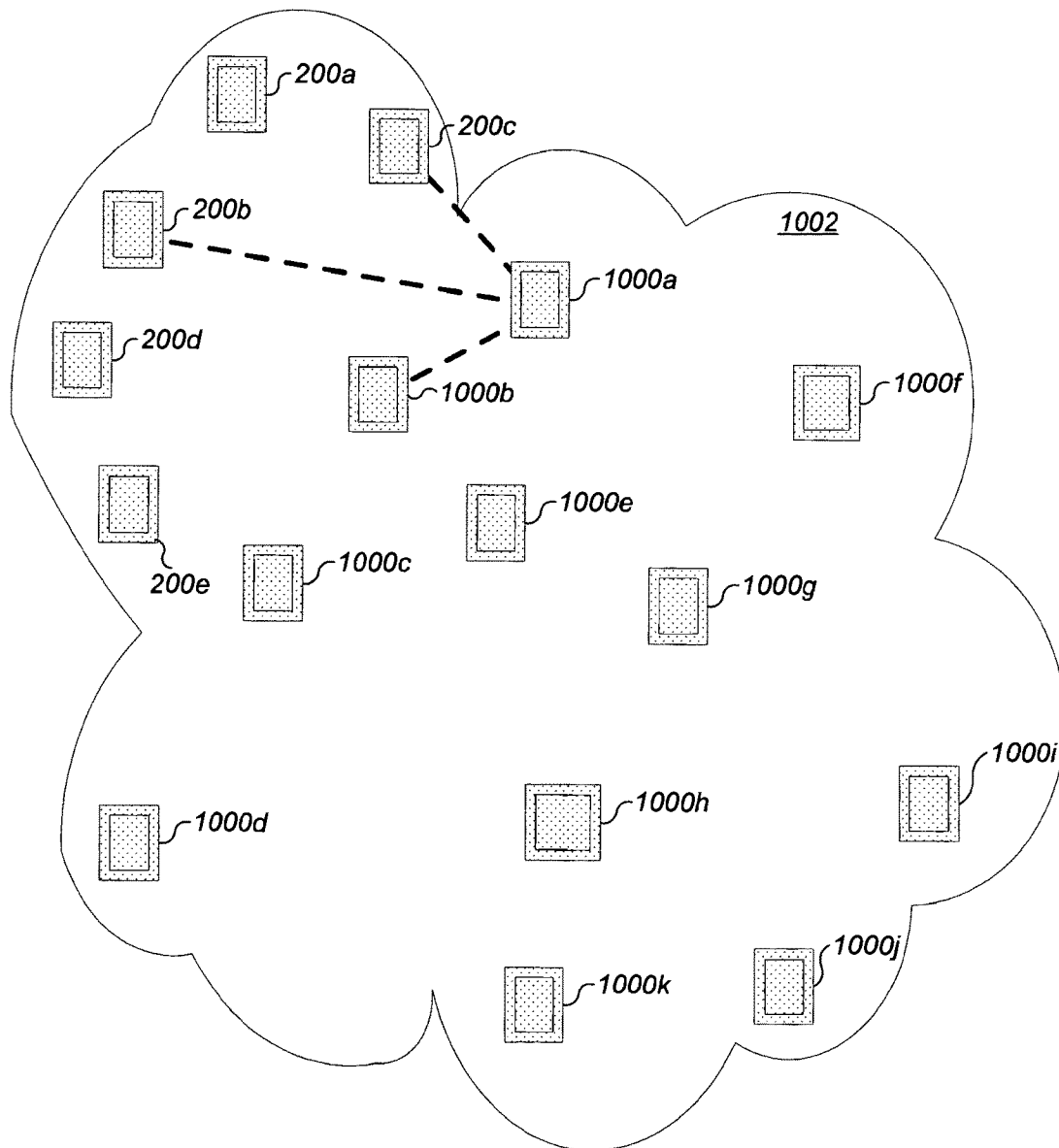


Fig. 10C

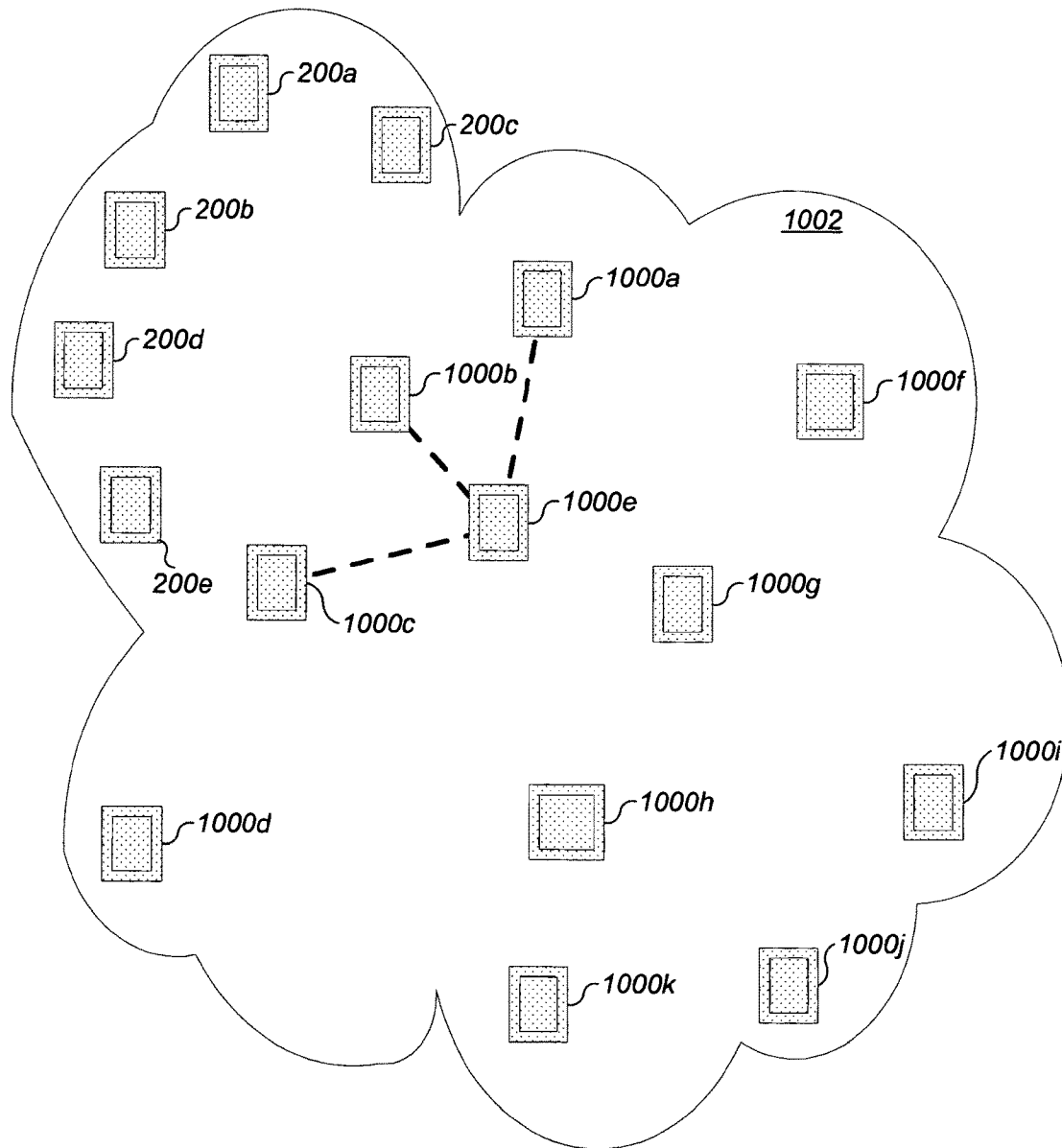


Fig. 10D

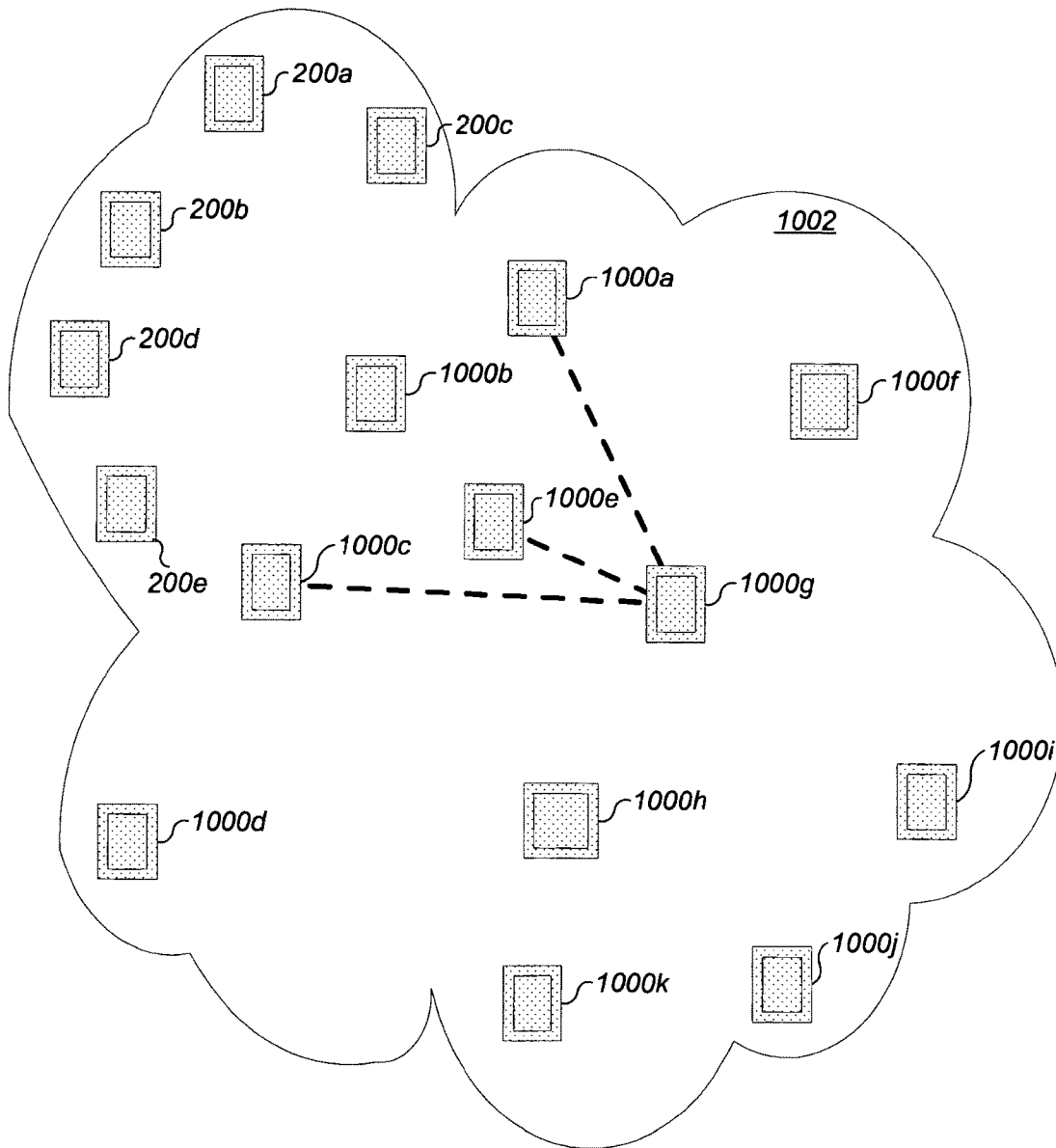


Fig. 10E

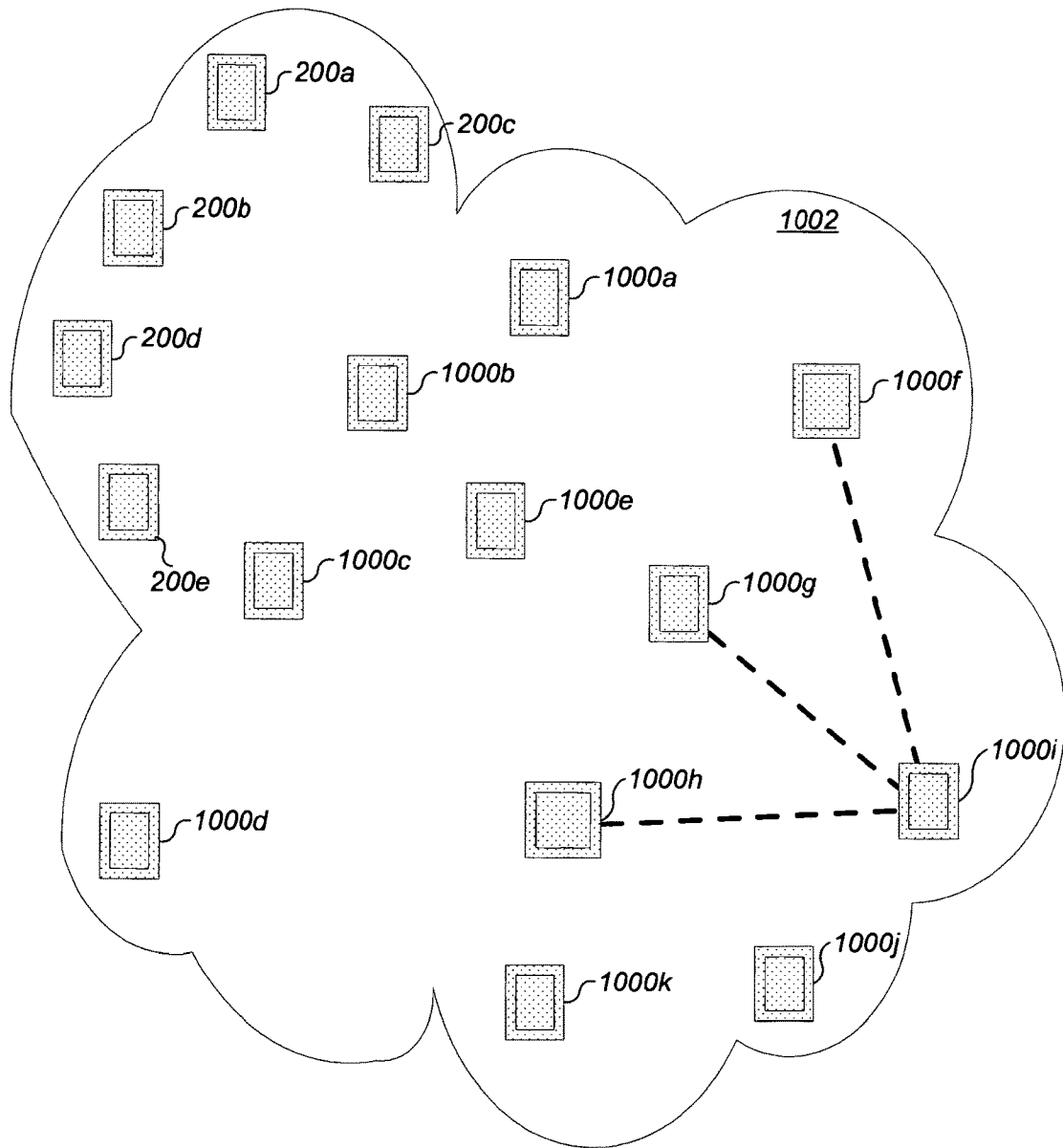


Fig. 10F

U.S. Patent

Nov. 19, 2019

Sheet 37 of 322

US 10,477,994 B2

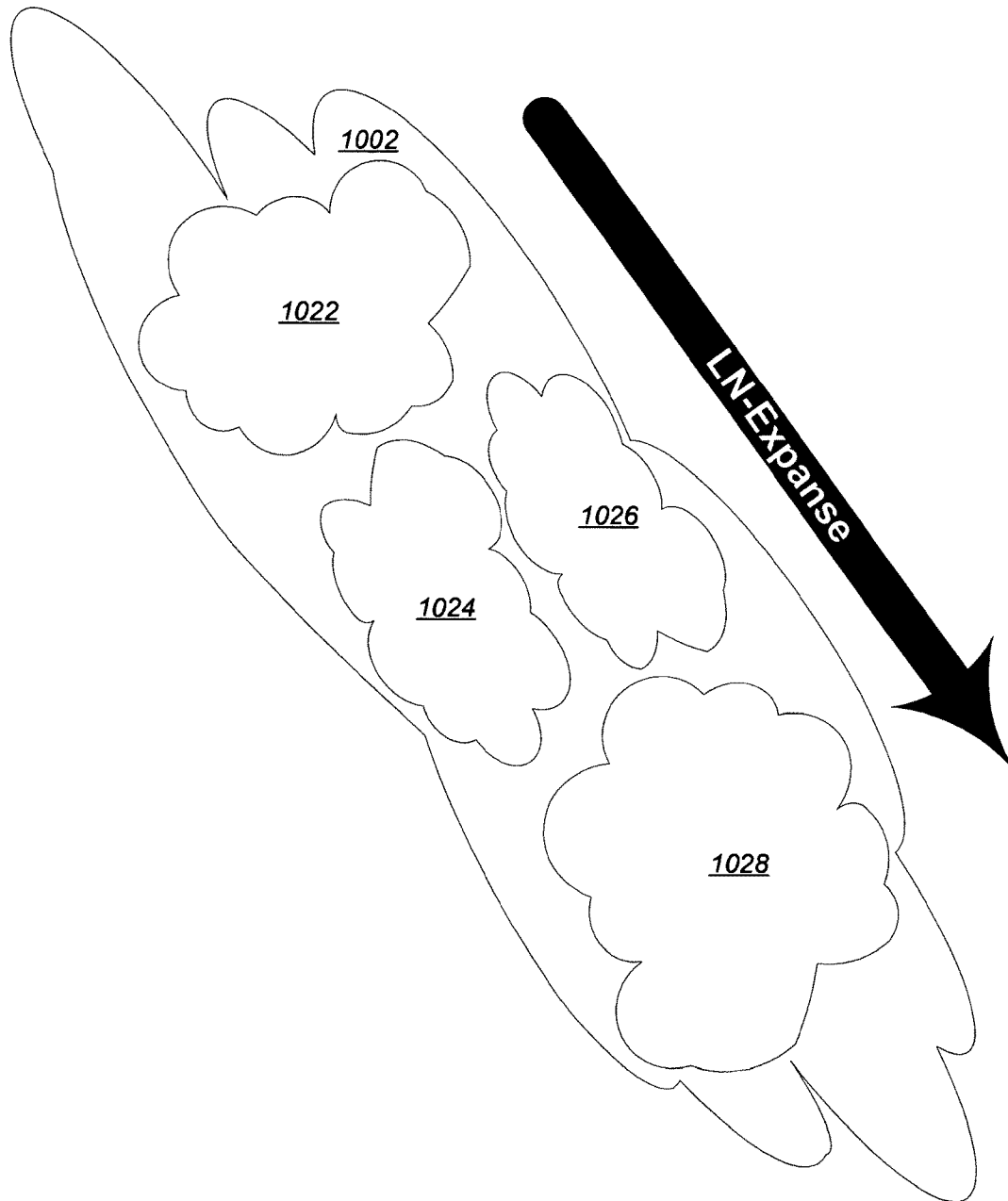


Fig. 10G

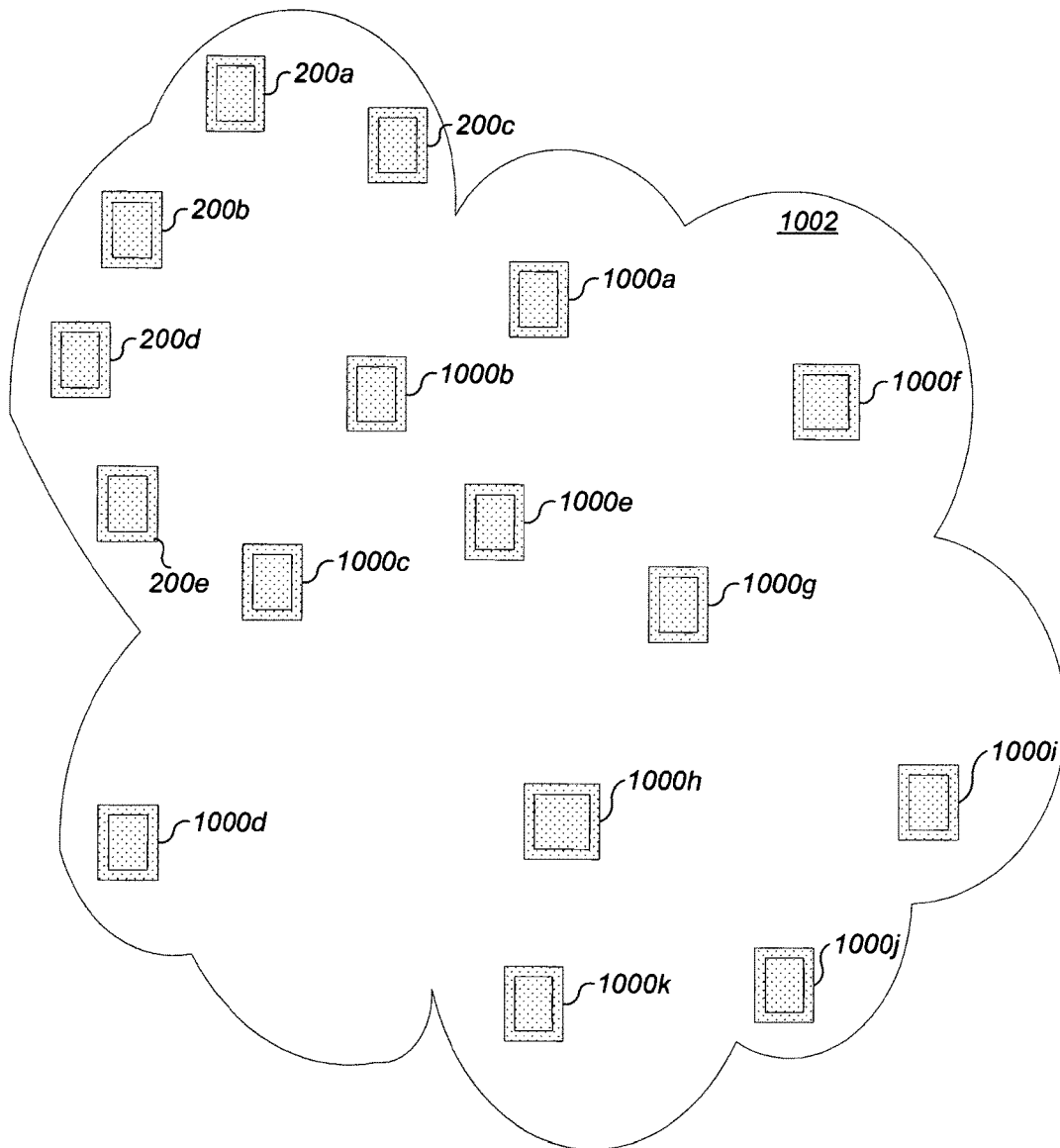


Fig. 10H

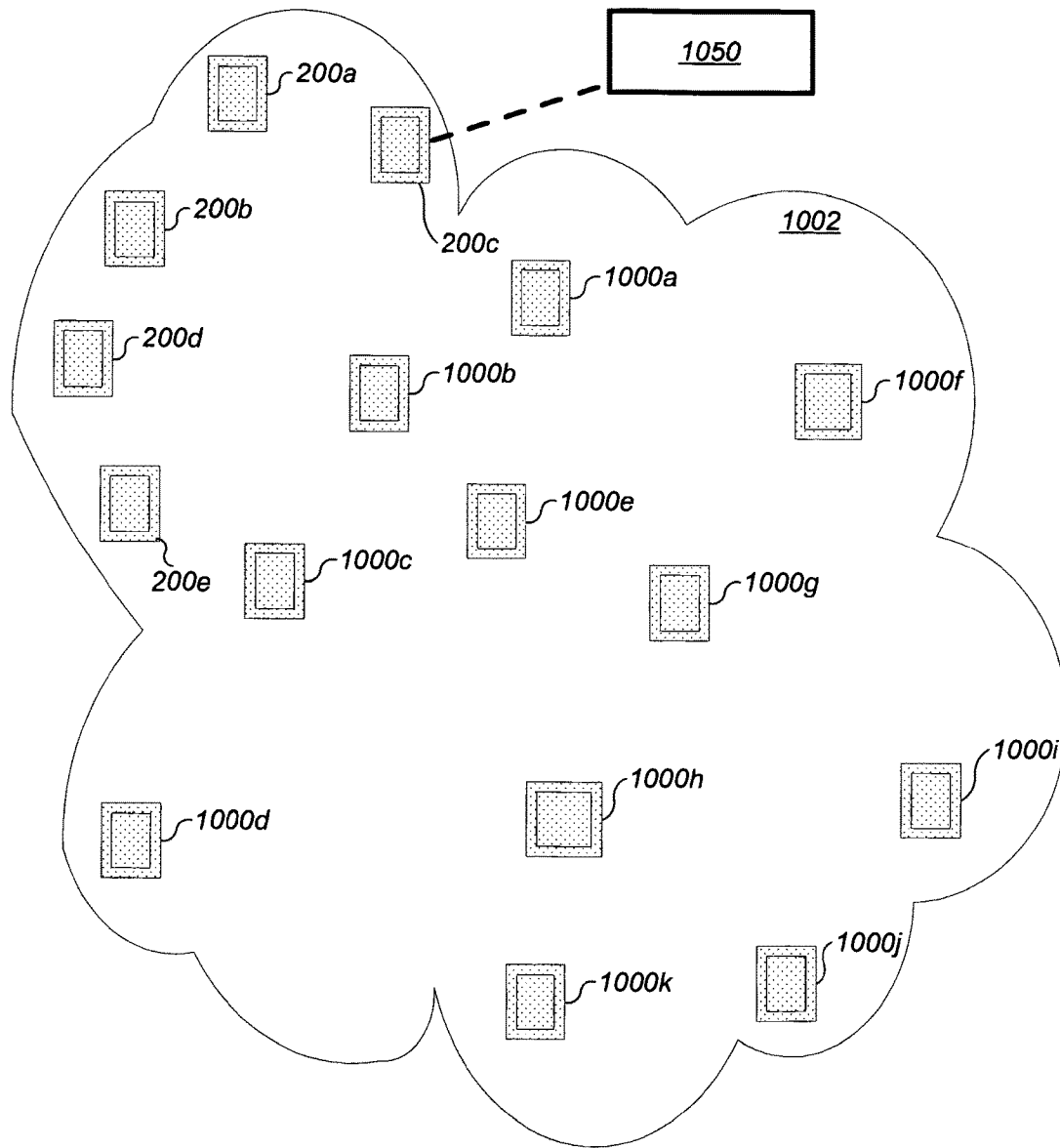
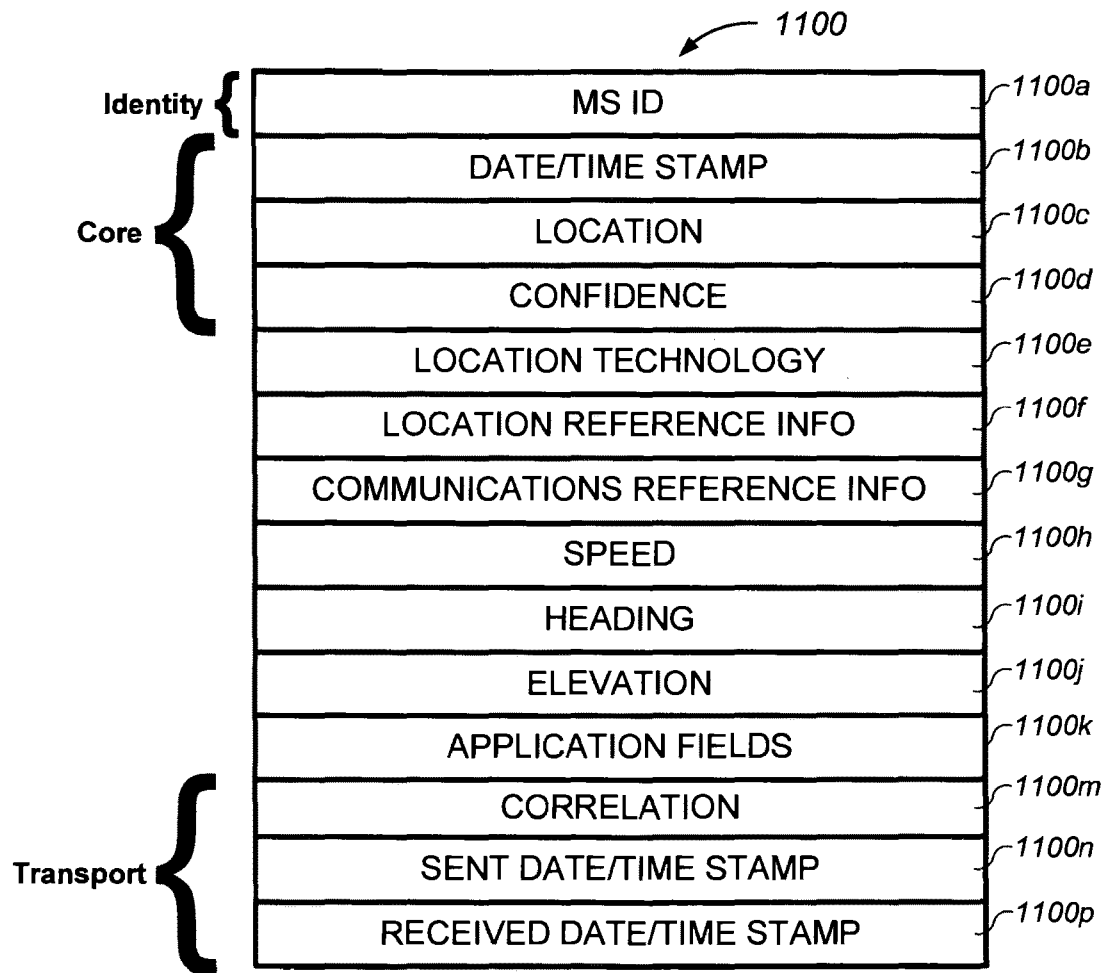


Fig. 10I

**Fig. 11A**

U.S. Patent

Nov. 19, 2019

Sheet 41 of 322

US 10,477,994 B2

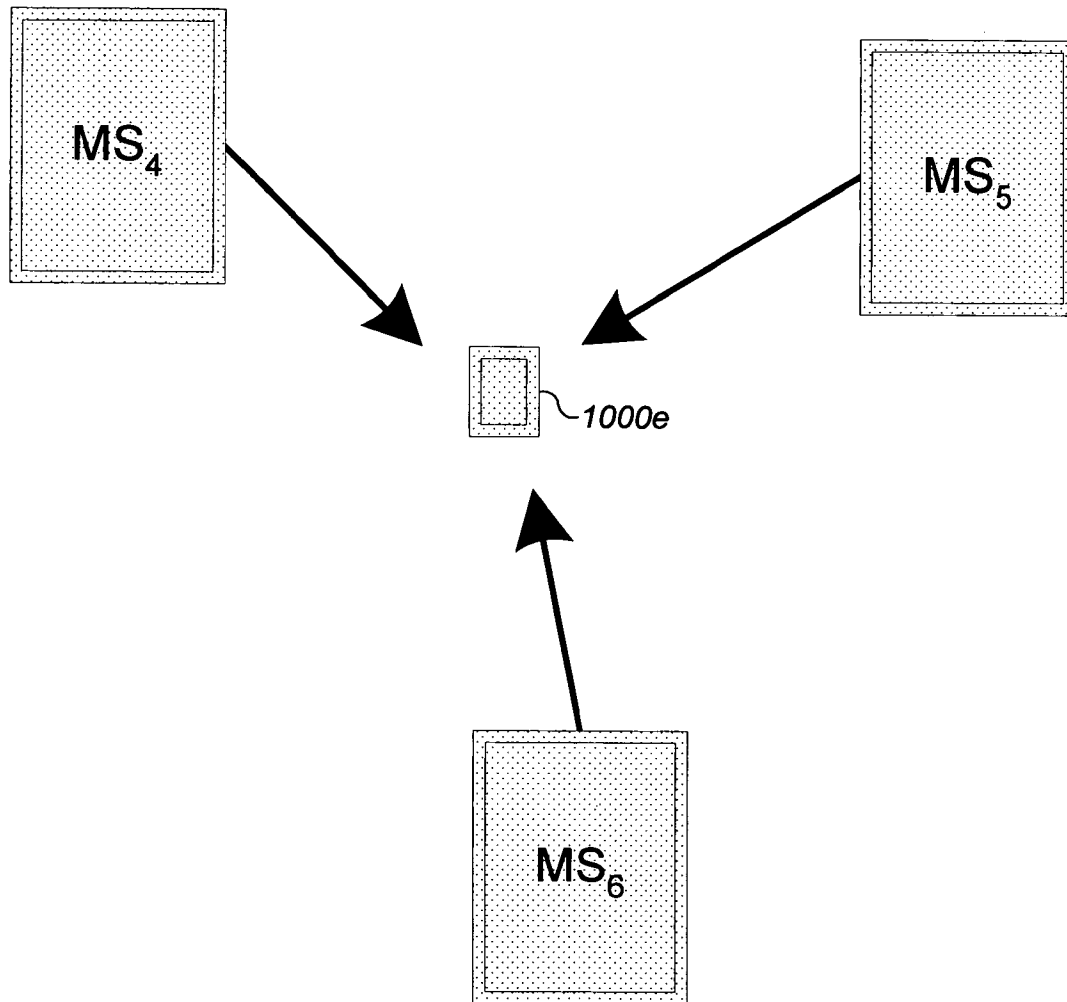


Fig. 11B

U.S. Patent

Nov. 19, 2019

Sheet 42 of 322

US 10,477,994 B2

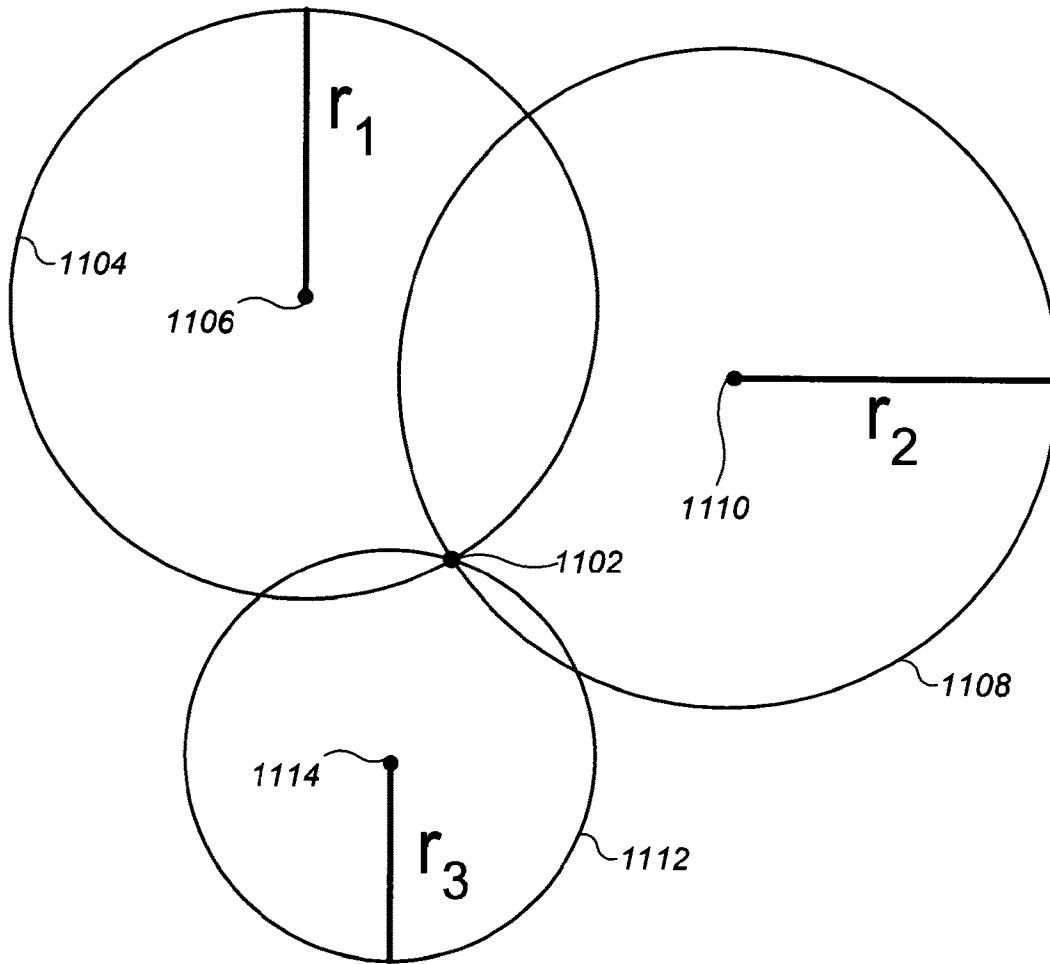


Fig. 11C

U.S. Patent

Nov. 19, 2019

Sheet 43 of 322

US 10,477,994 B2

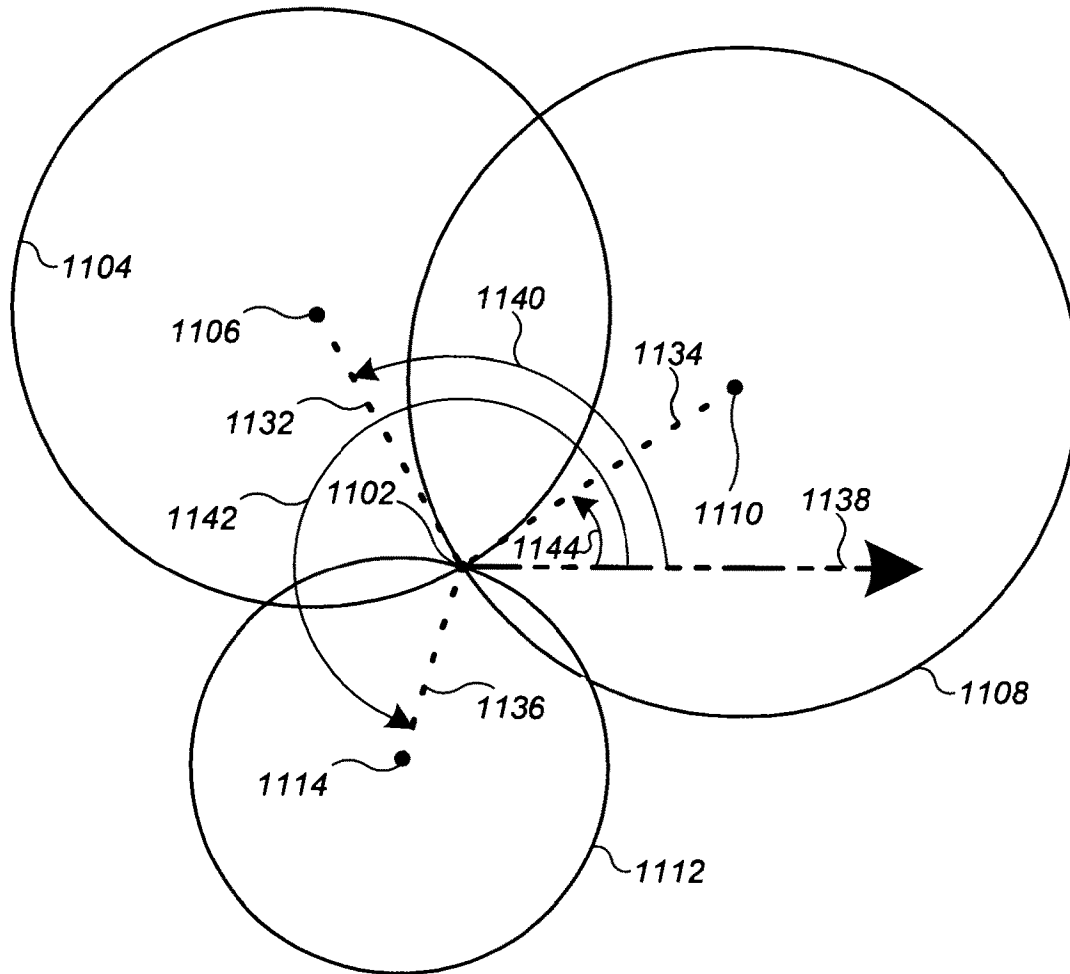


Fig. 11D

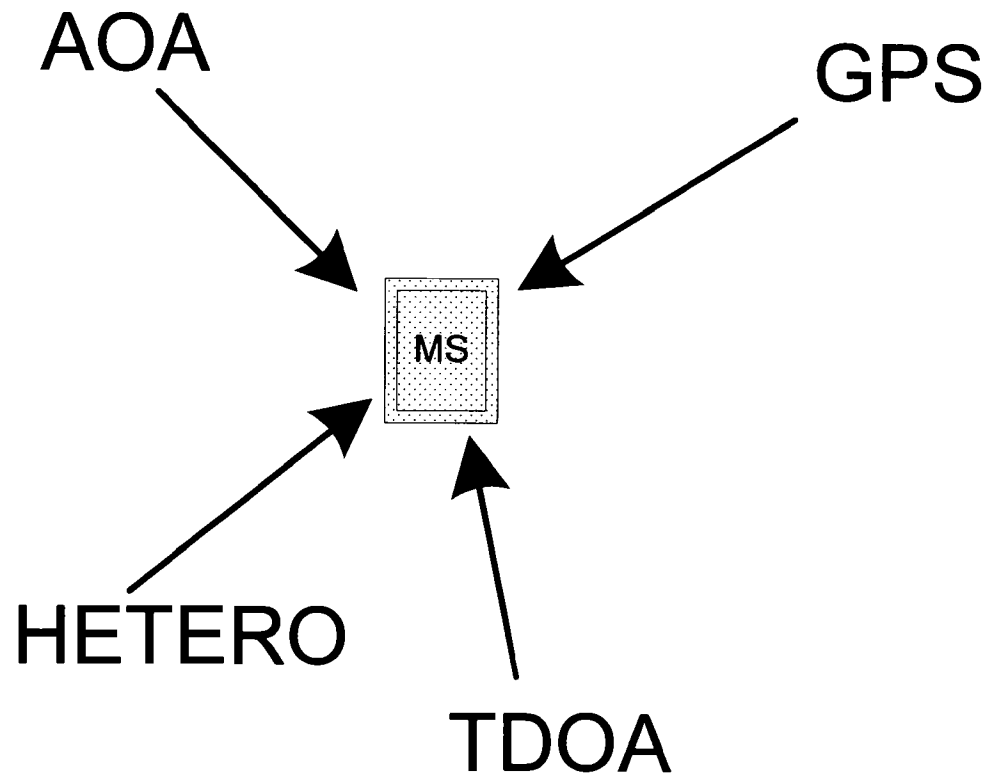


Fig. 11E

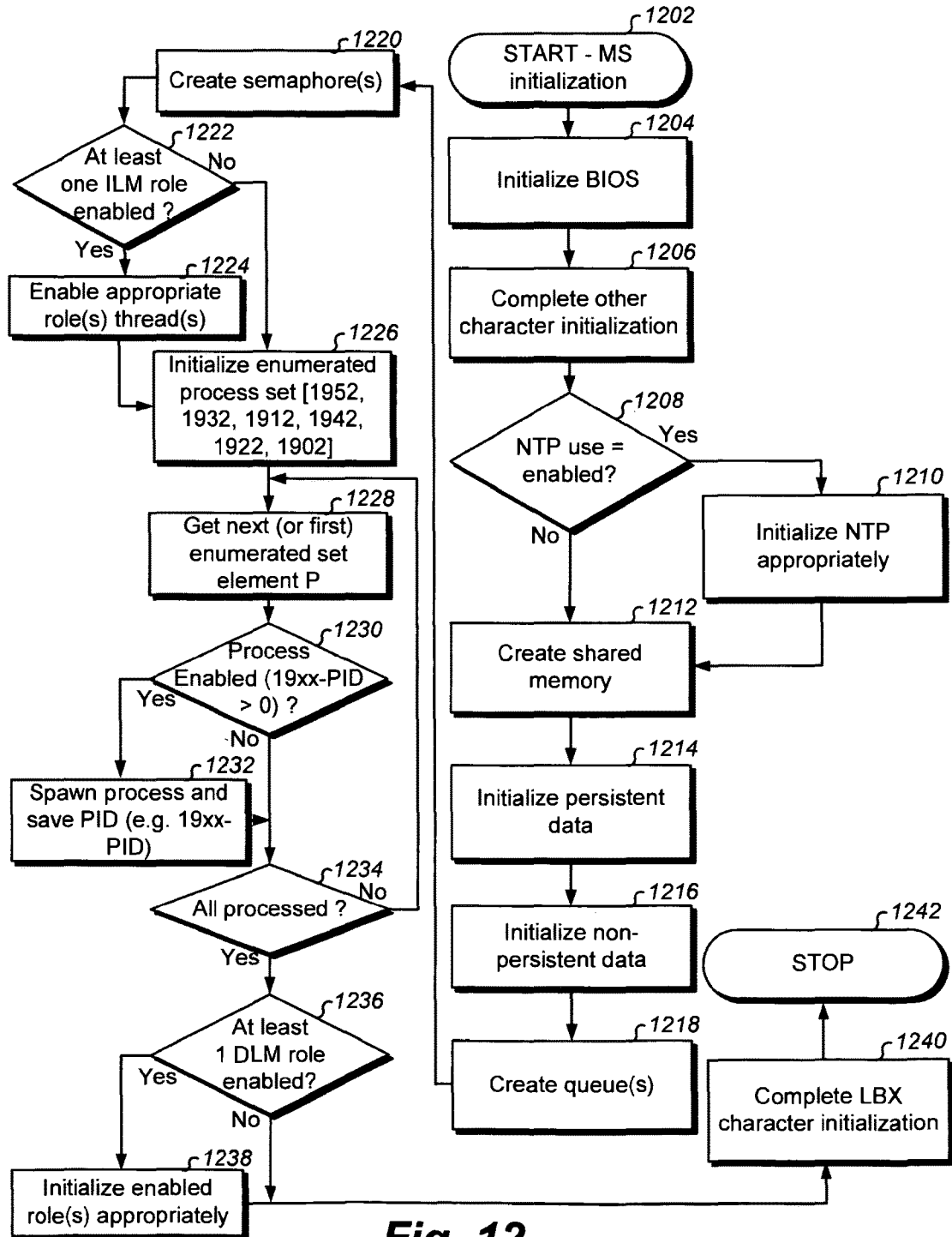


Fig. 12

U.S. Patent

Nov. 19, 2019

Sheet 46 of 322

US 10,477,994 B2

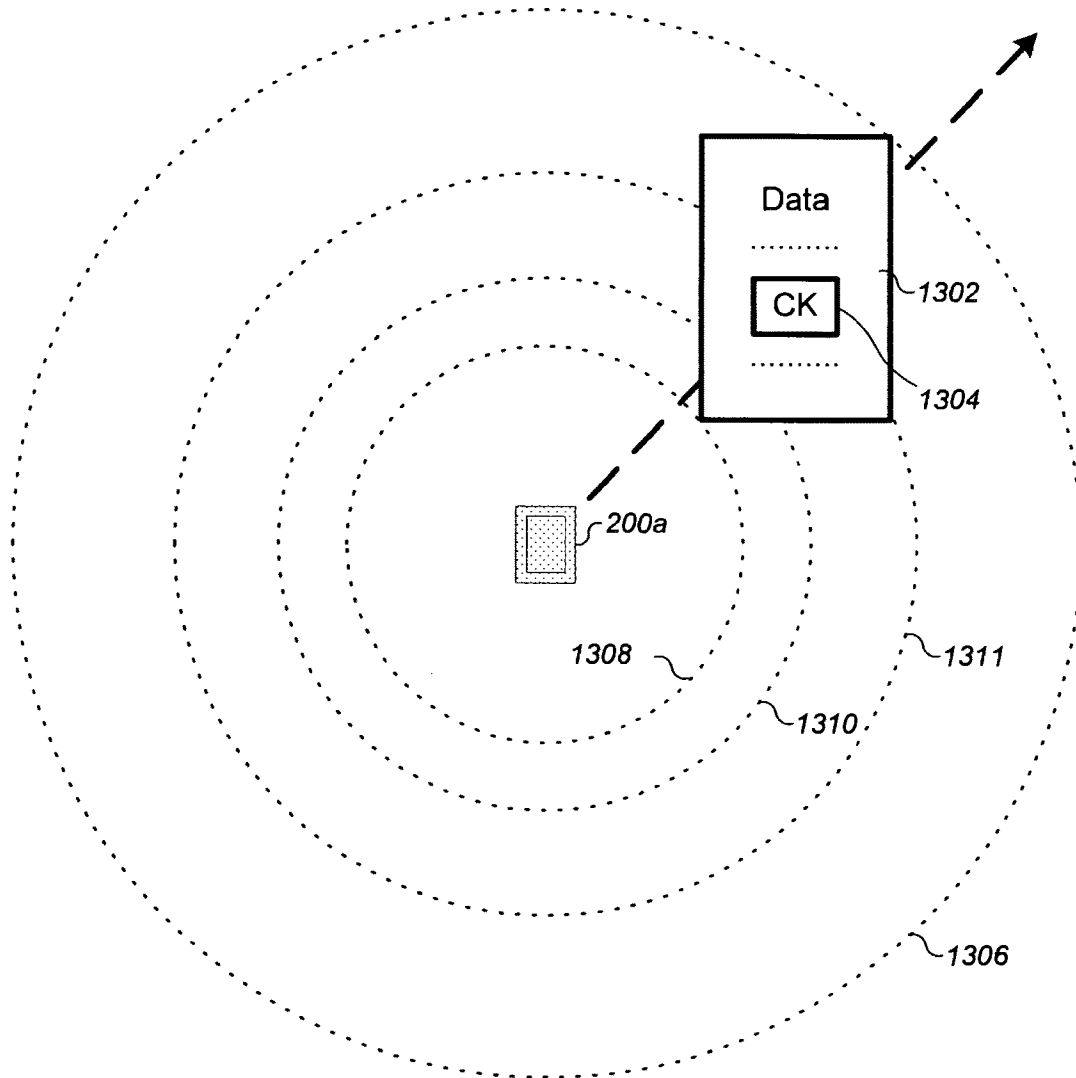


Fig. 13A

U.S. Patent

Nov. 19, 2019

Sheet 47 of 322

US 10,477,994 B2

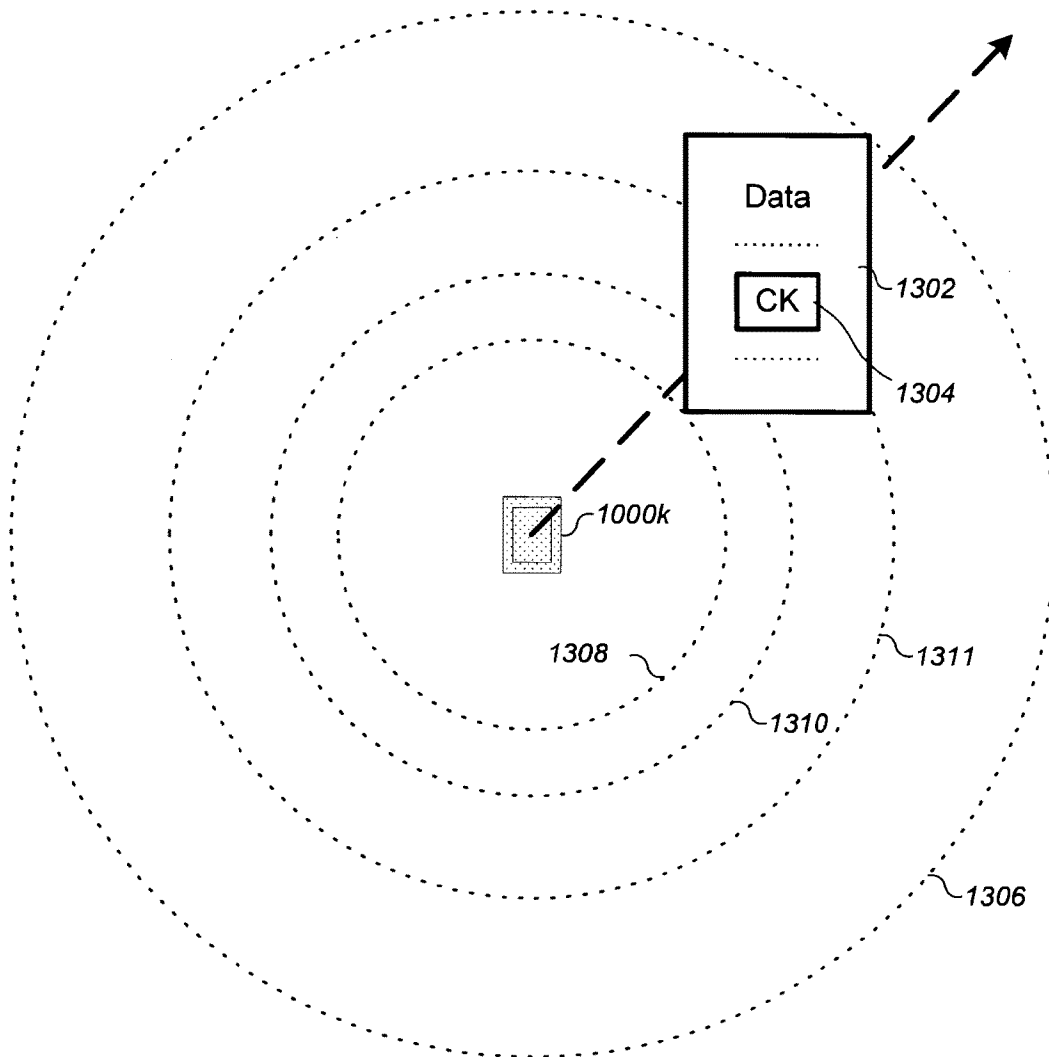


Fig. 13B

U.S. Patent

Nov. 19, 2019

Sheet 48 of 322

US 10,477,994 B2

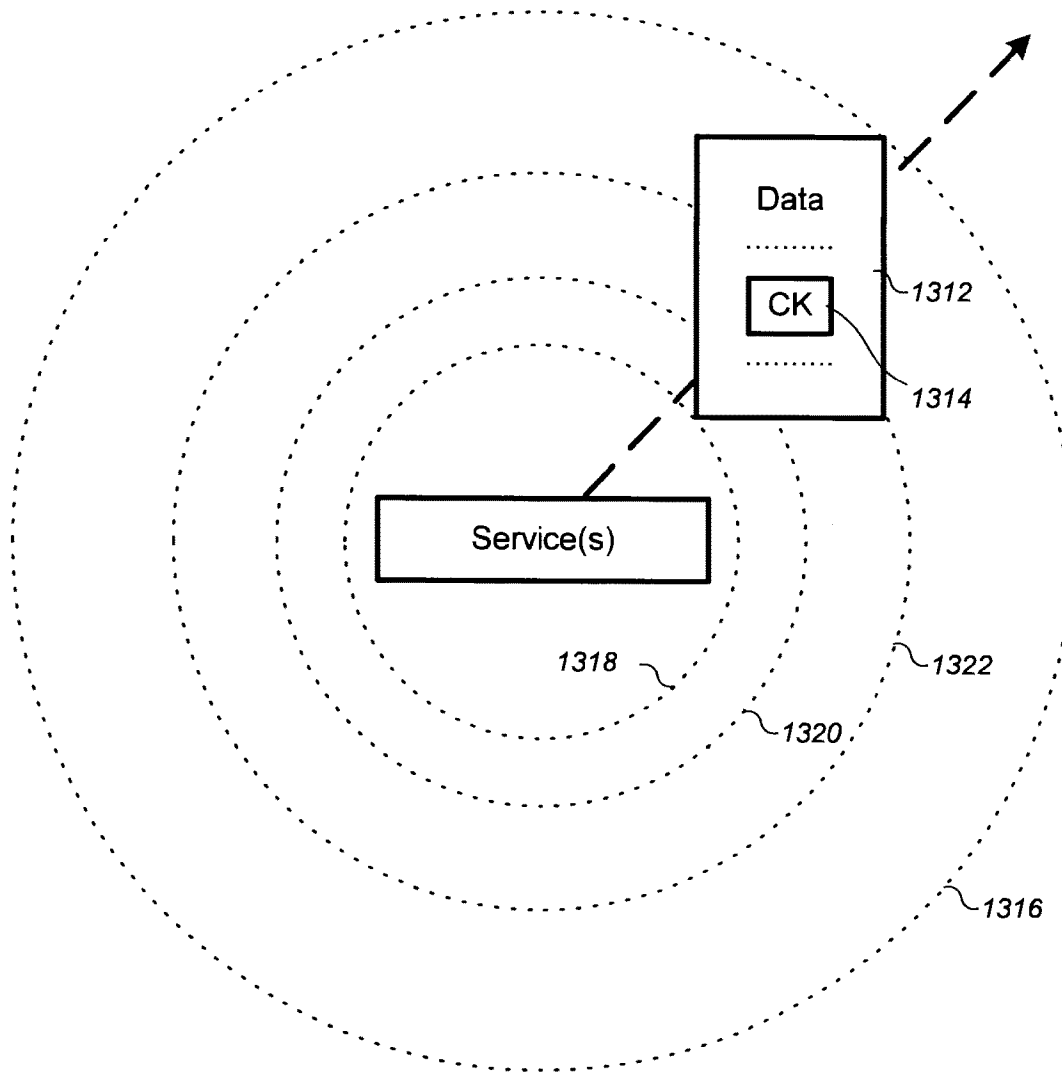


Fig. 13C

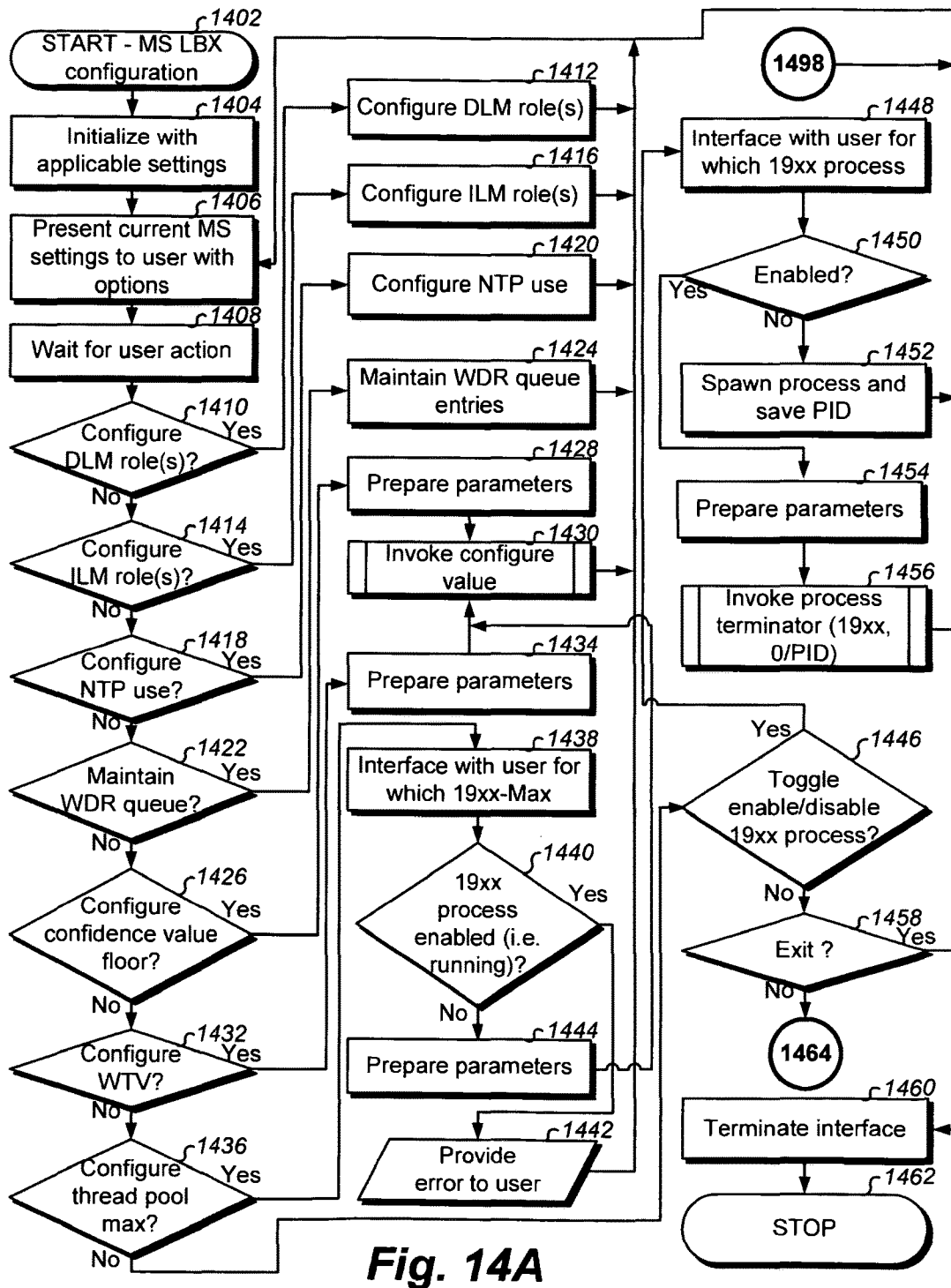
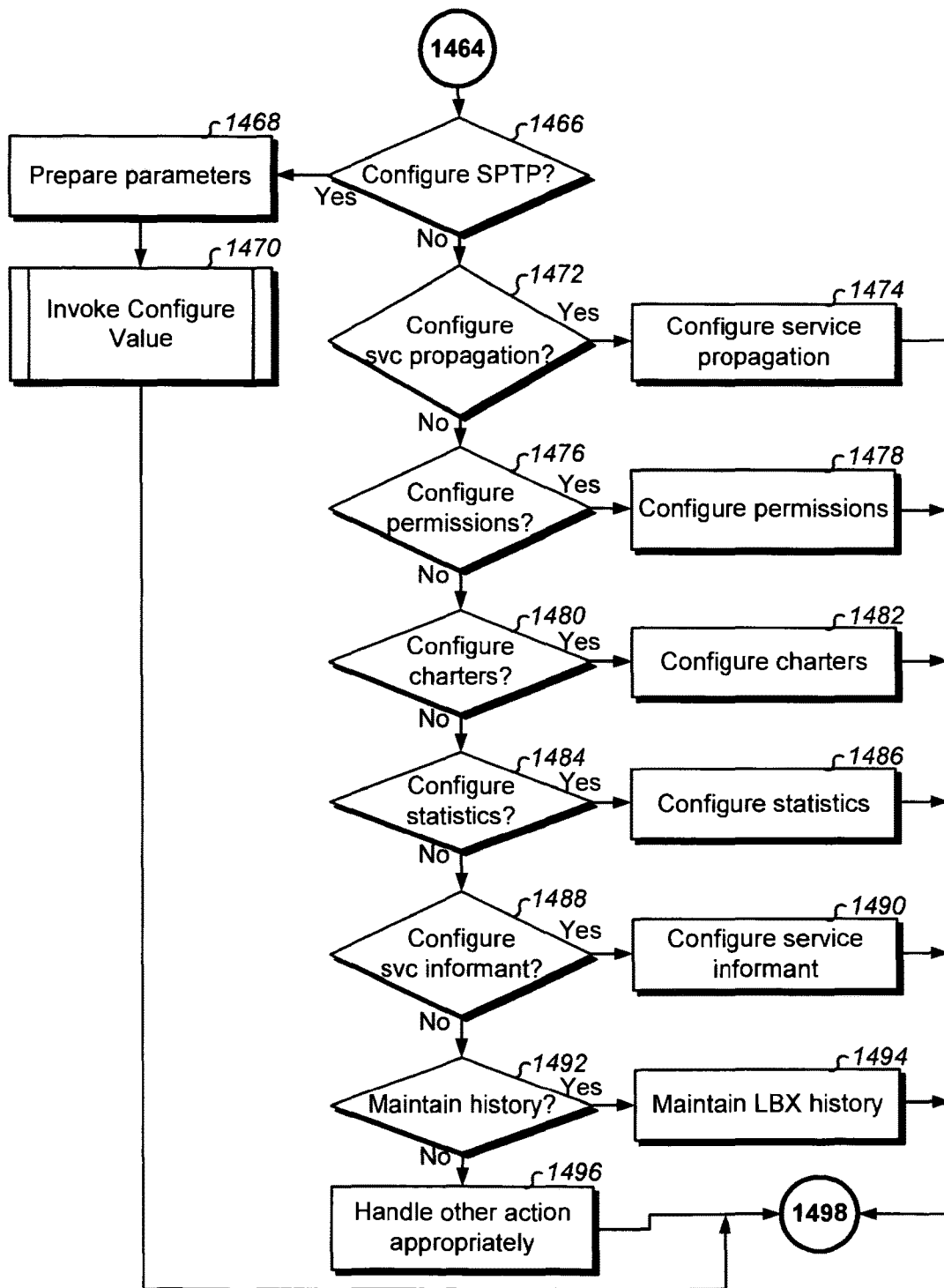
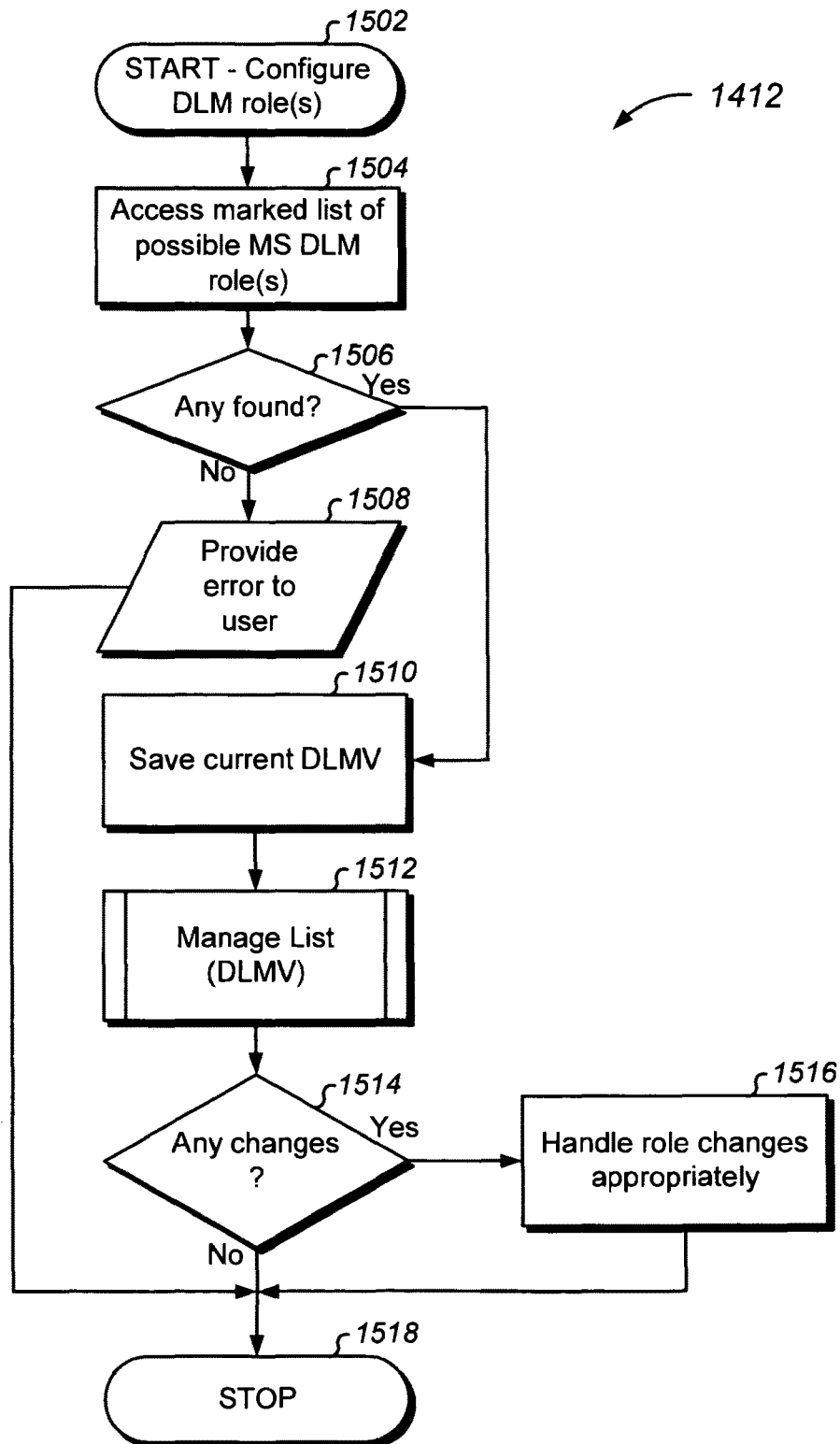
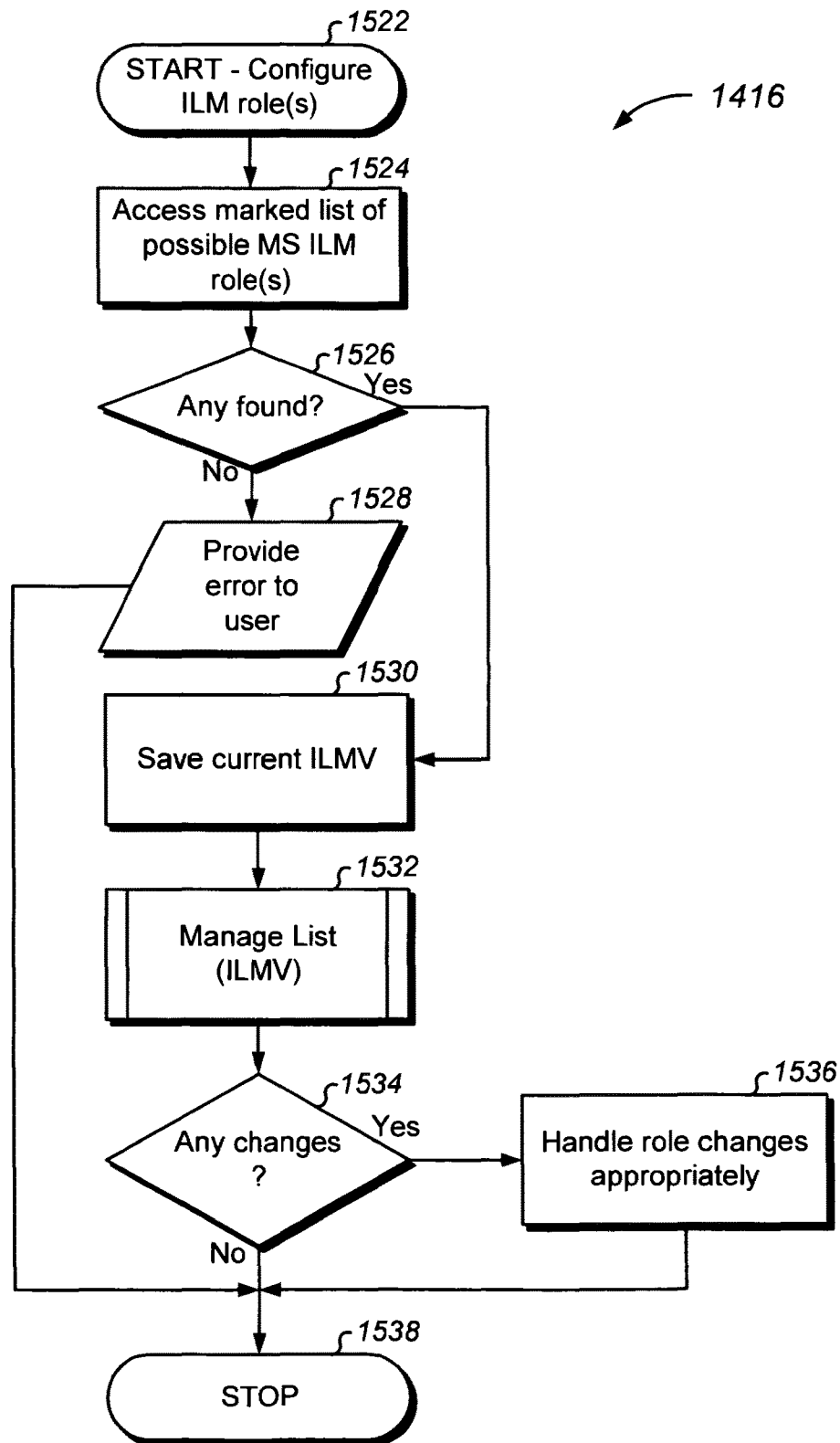
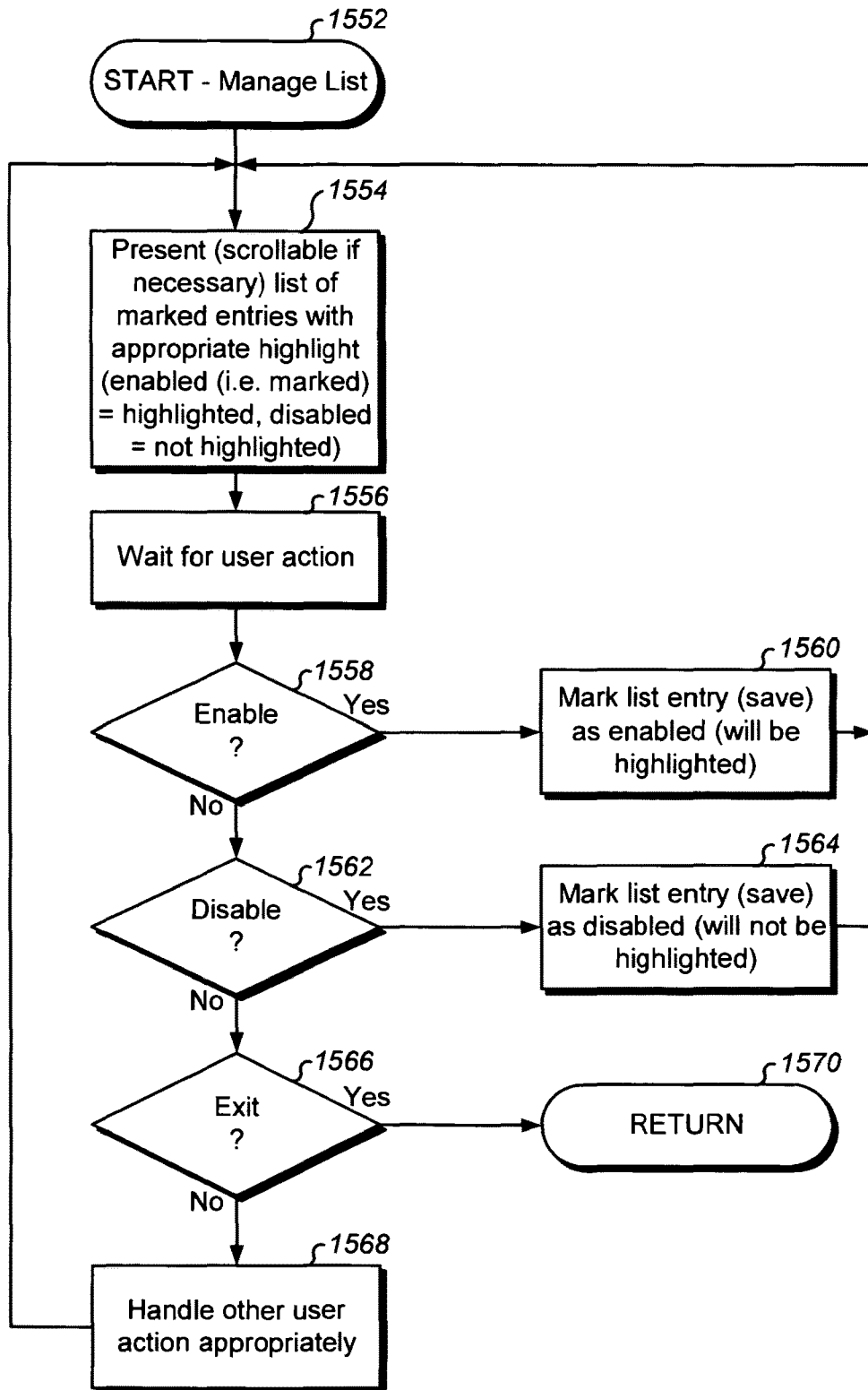


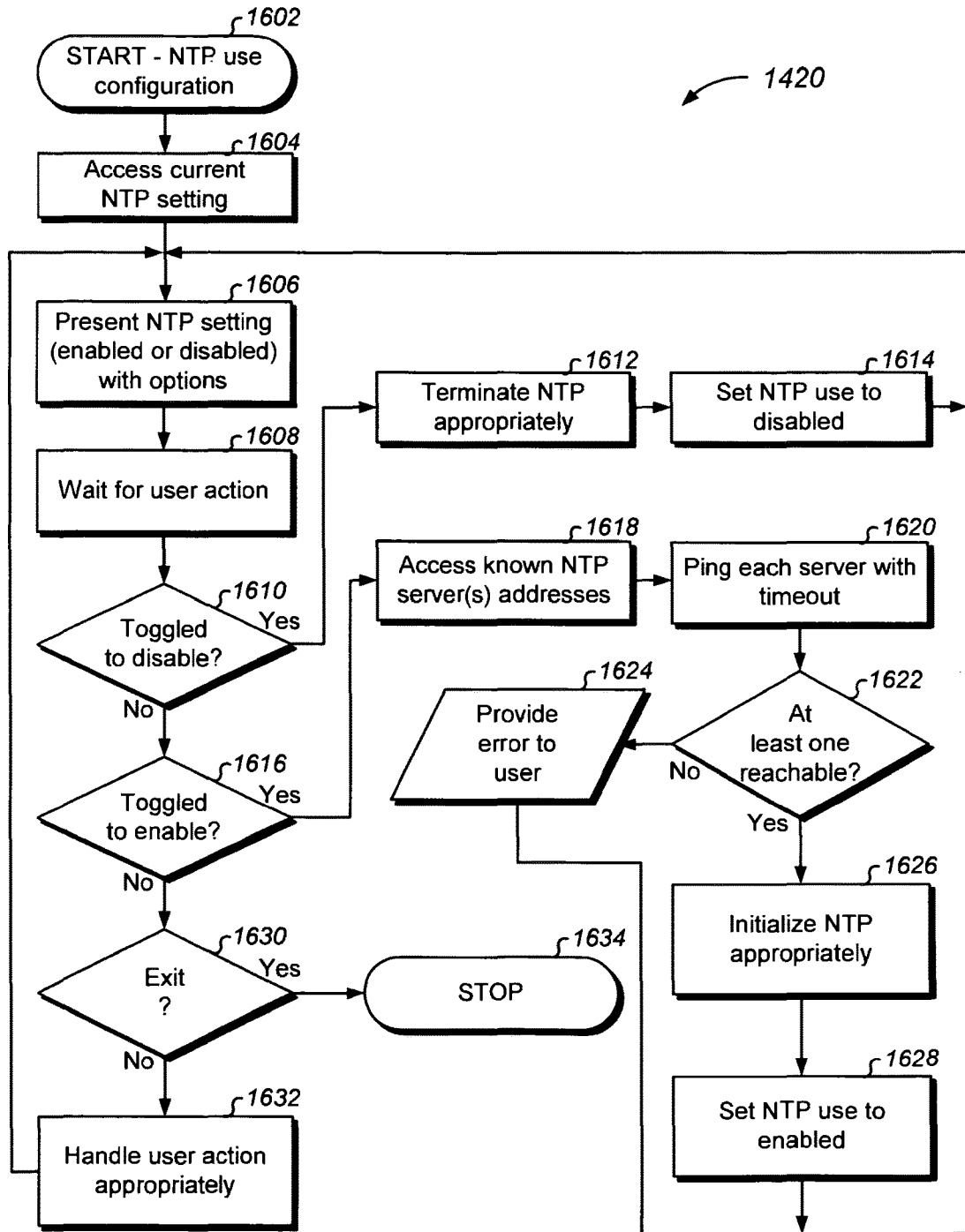
Fig. 14A

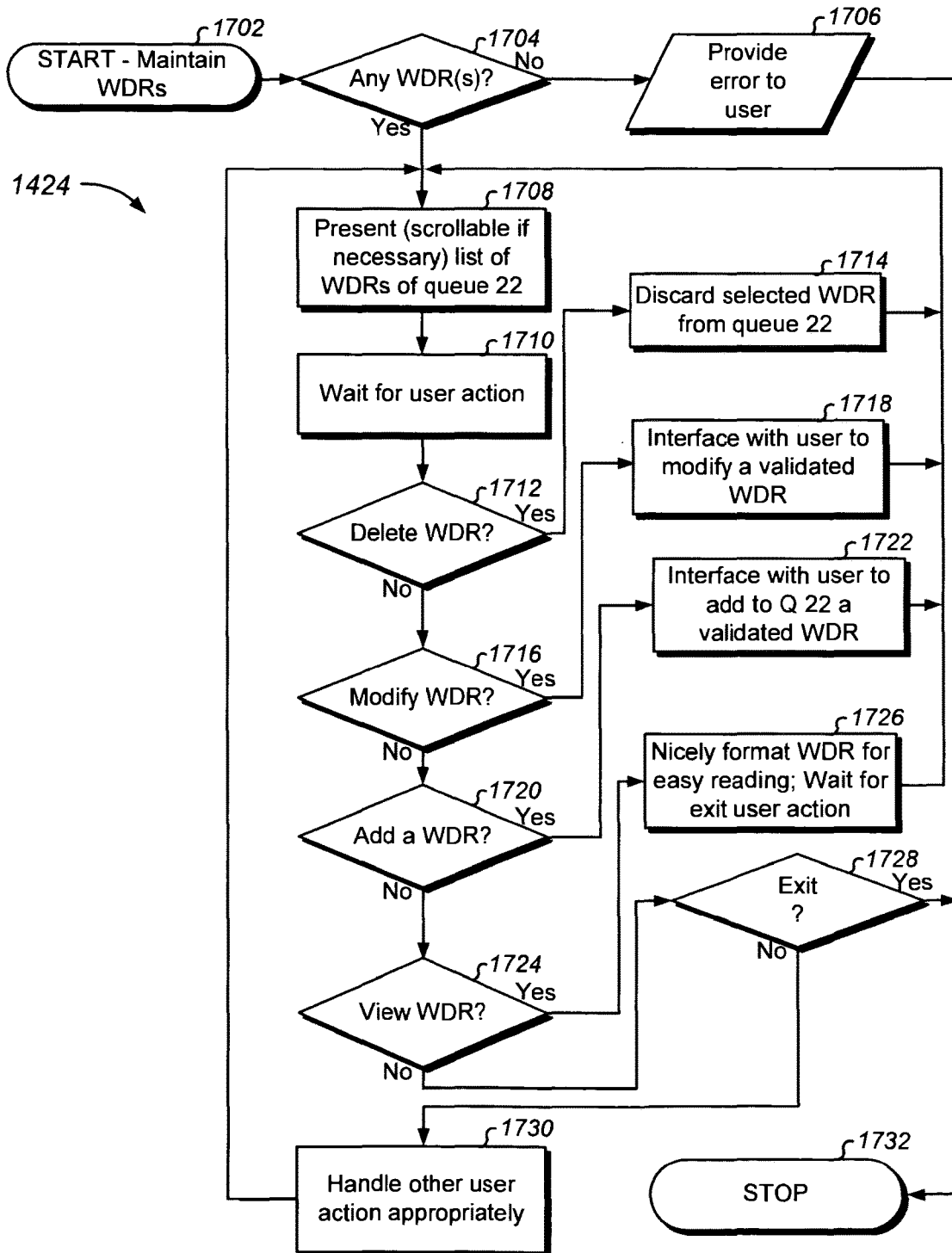
**Fig. 14B**

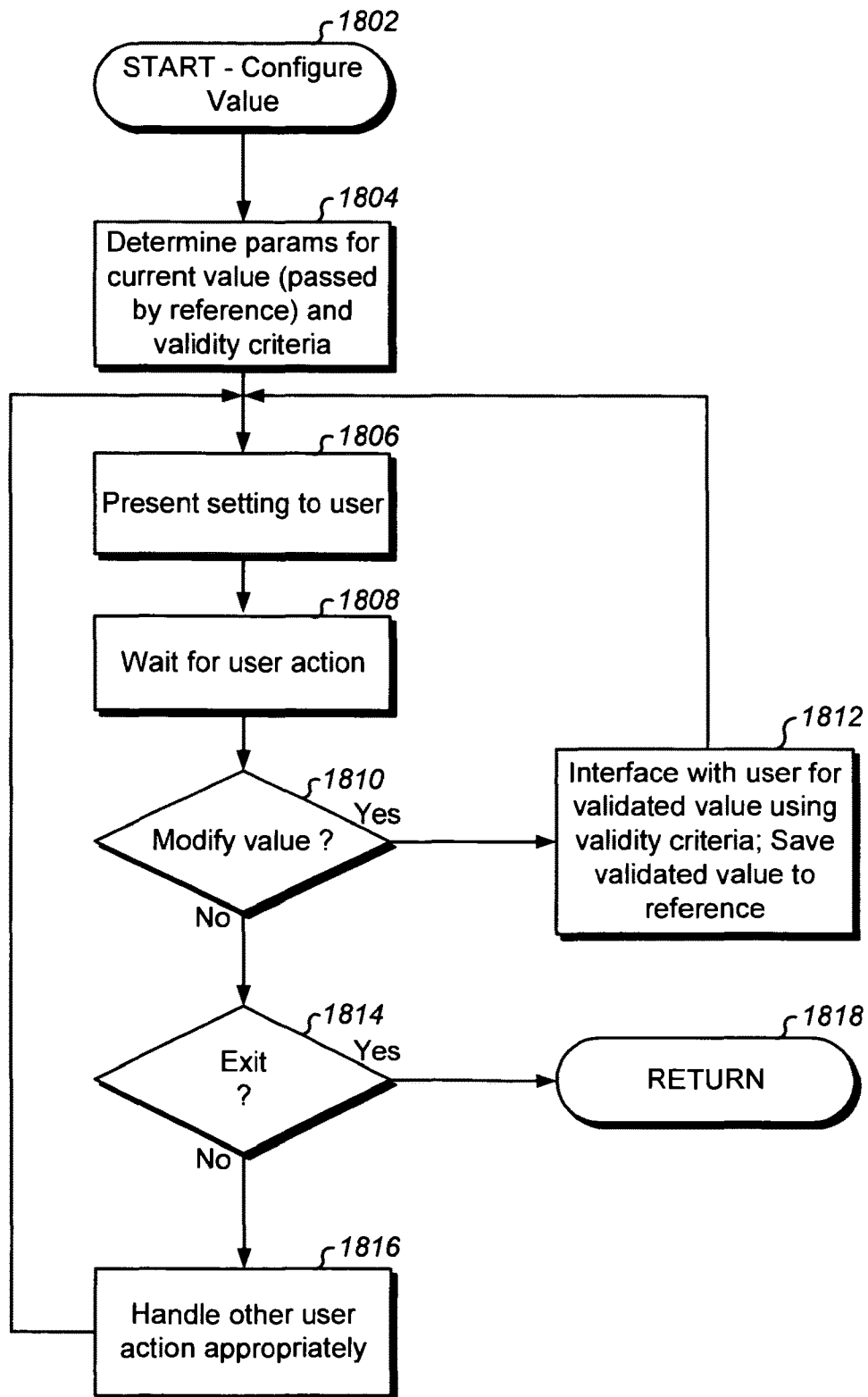
**Fig. 15A**

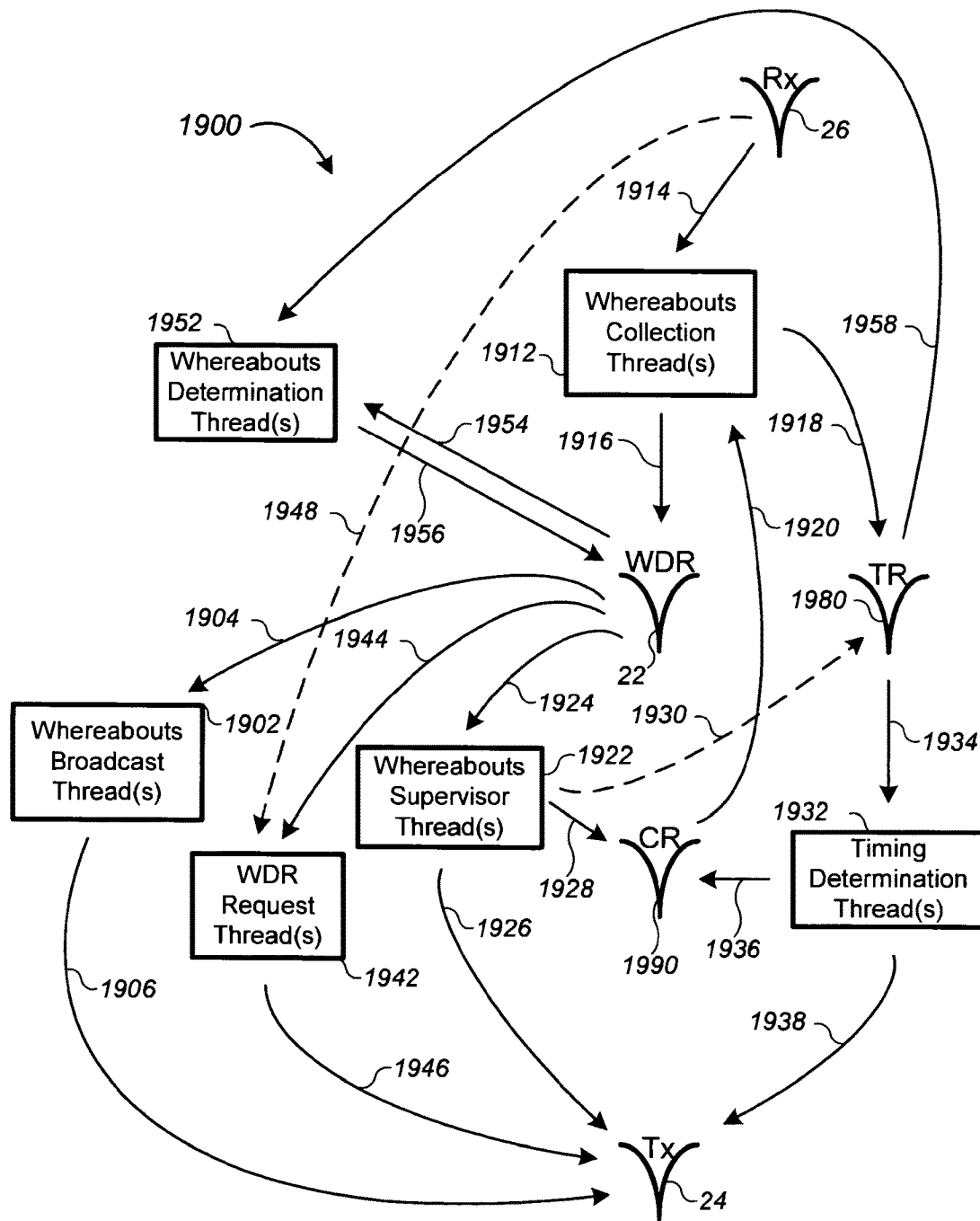
**Fig. 15B**

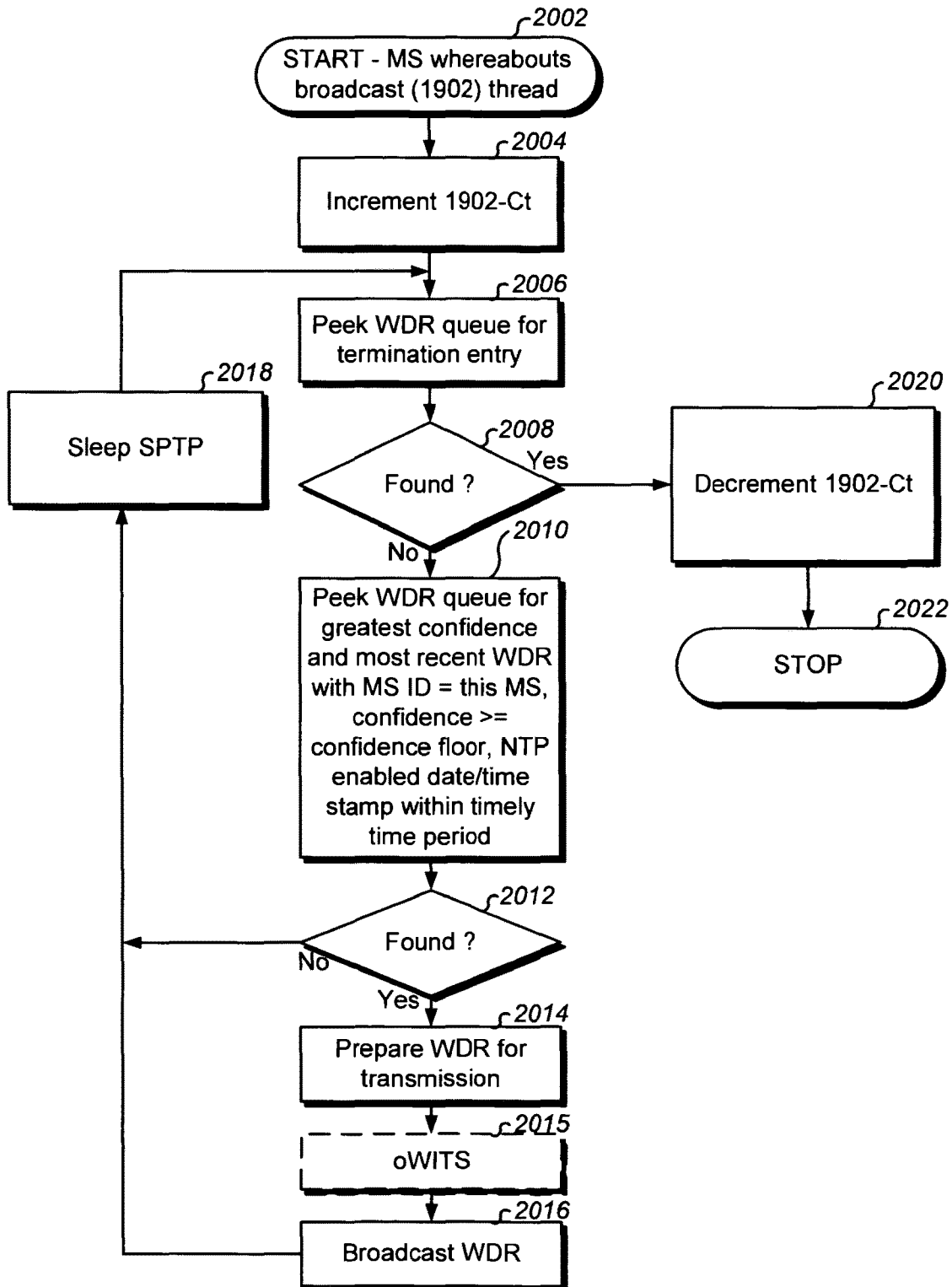
**Fig. 15C**

**Fig. 16**

**Fig. 17**

**Fig. 18**

**Fig. 19**

**Fig. 20**

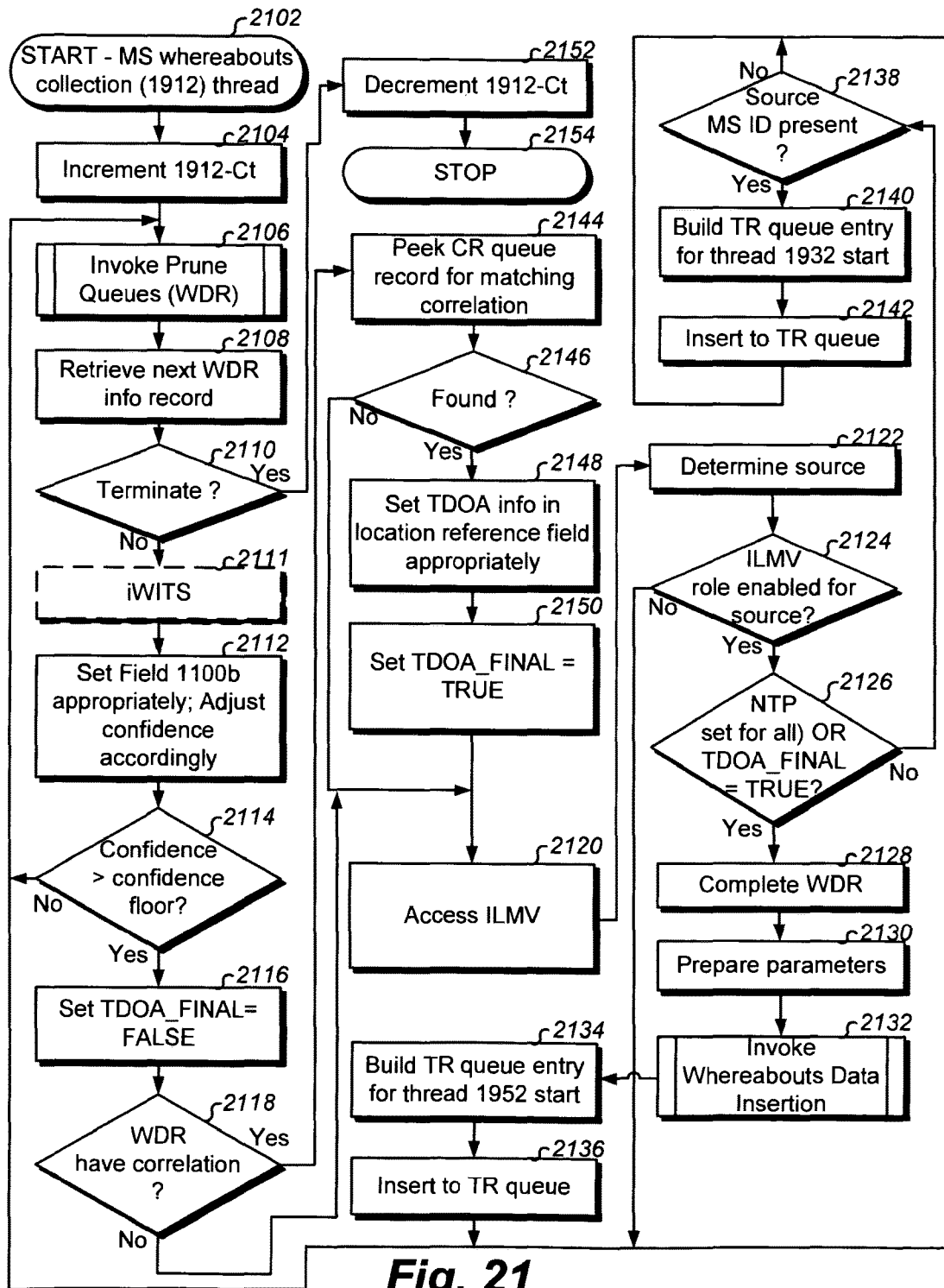
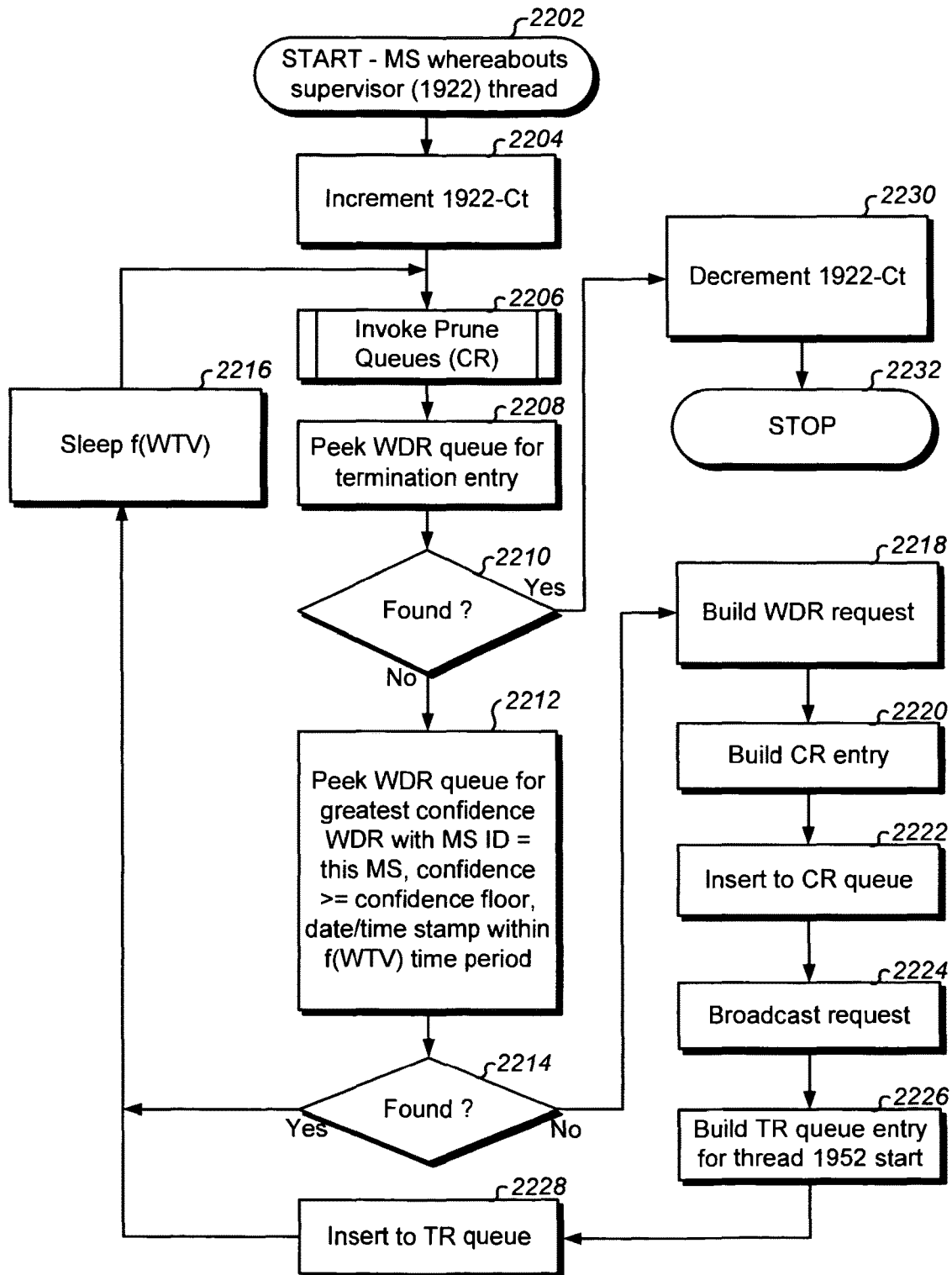
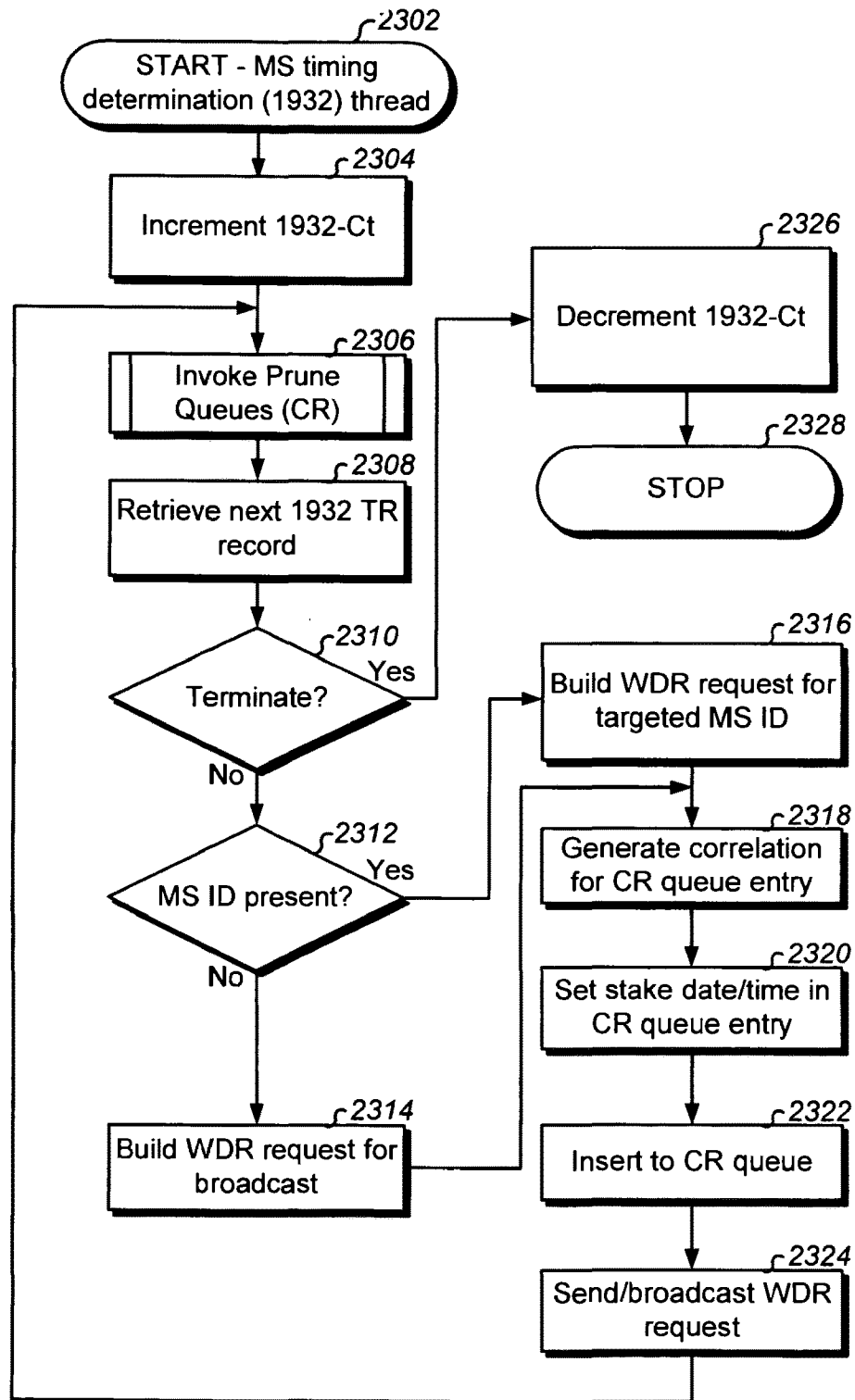


Fig. 21

**Fig. 22**

**Fig. 23**

U.S. Patent

Nov. 19, 2019

Sheet 62 of 322

US 10,477,994 B2

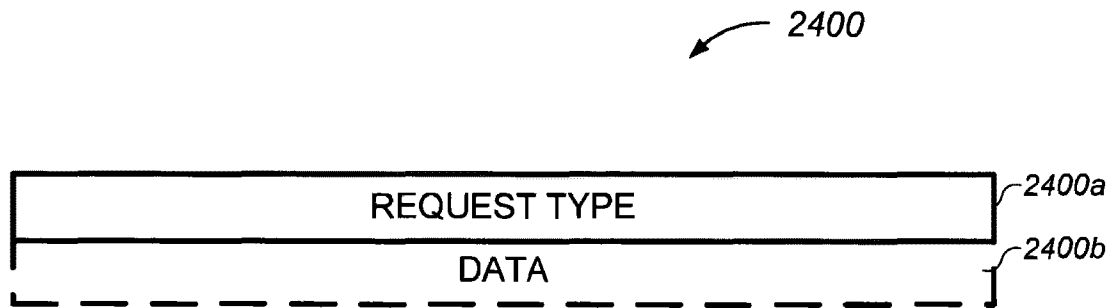


Fig. 24A

U.S. Patent

Nov. 19, 2019

Sheet 63 of 322

US 10,477,994 B2

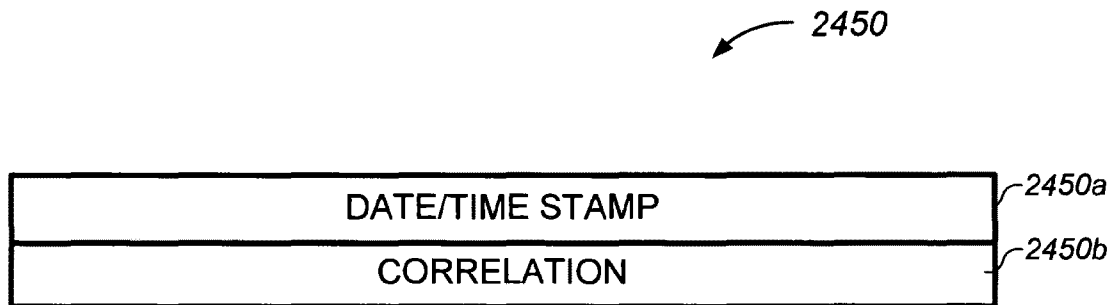


Fig. 24B

U.S. Patent

Nov. 19, 2019

Sheet 64 of 322

US 10,477,994 B2

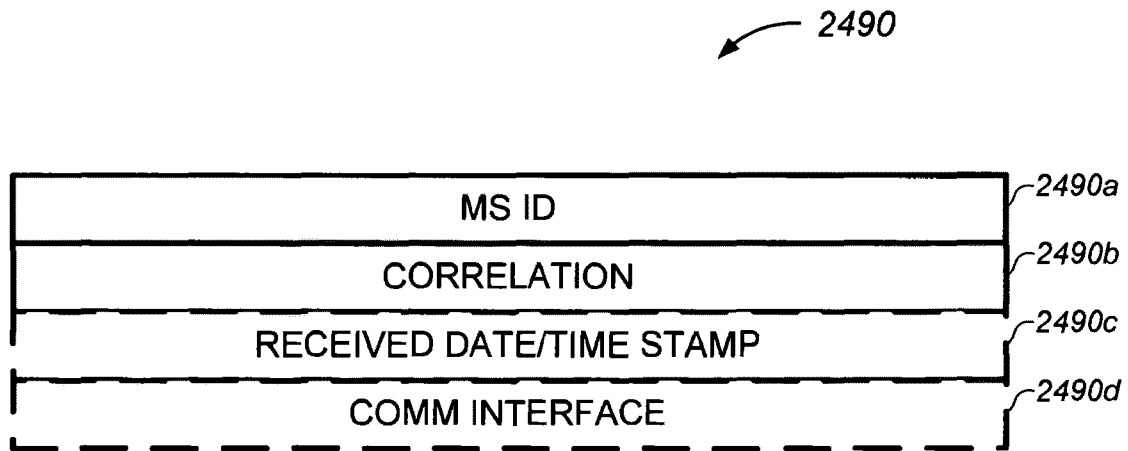
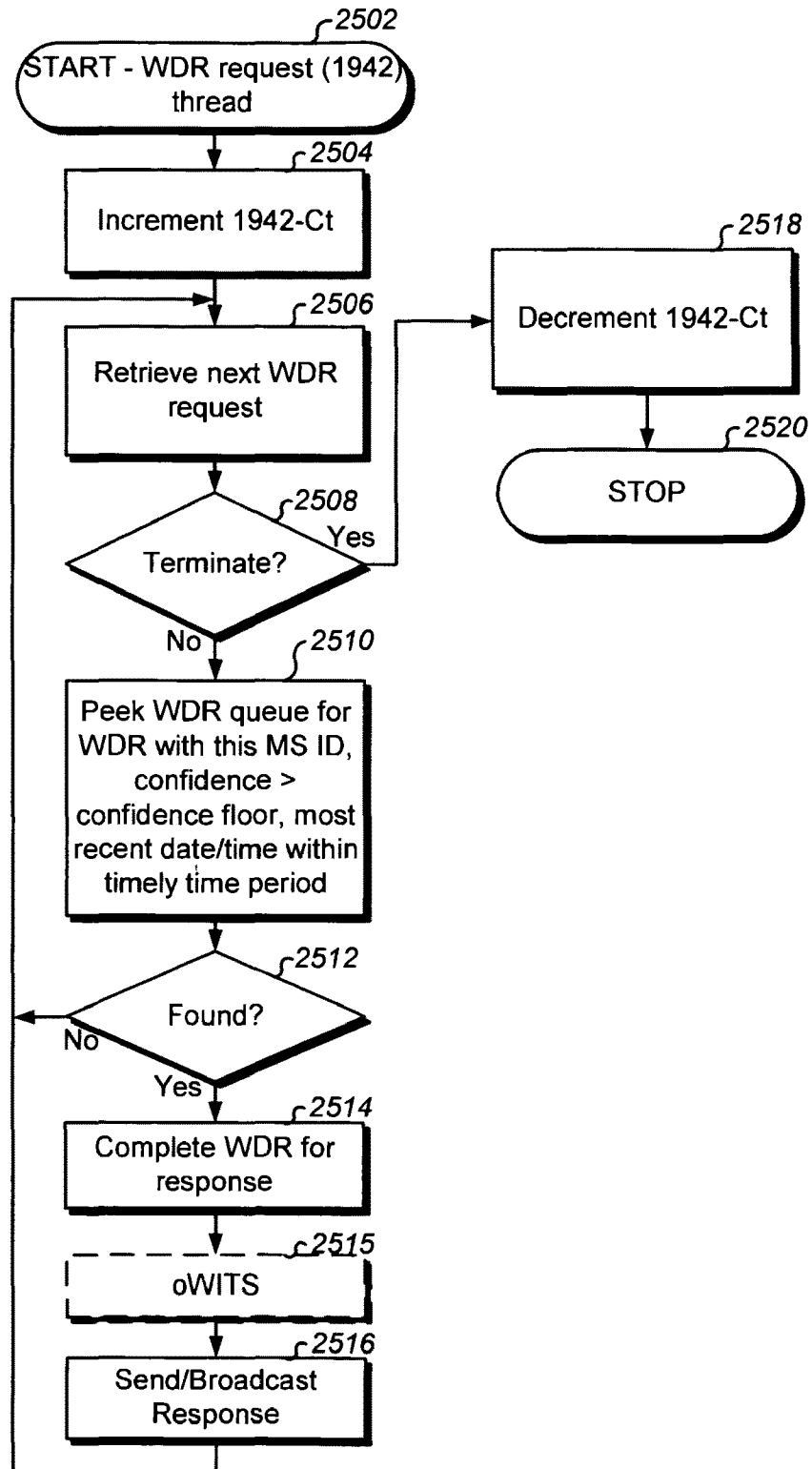
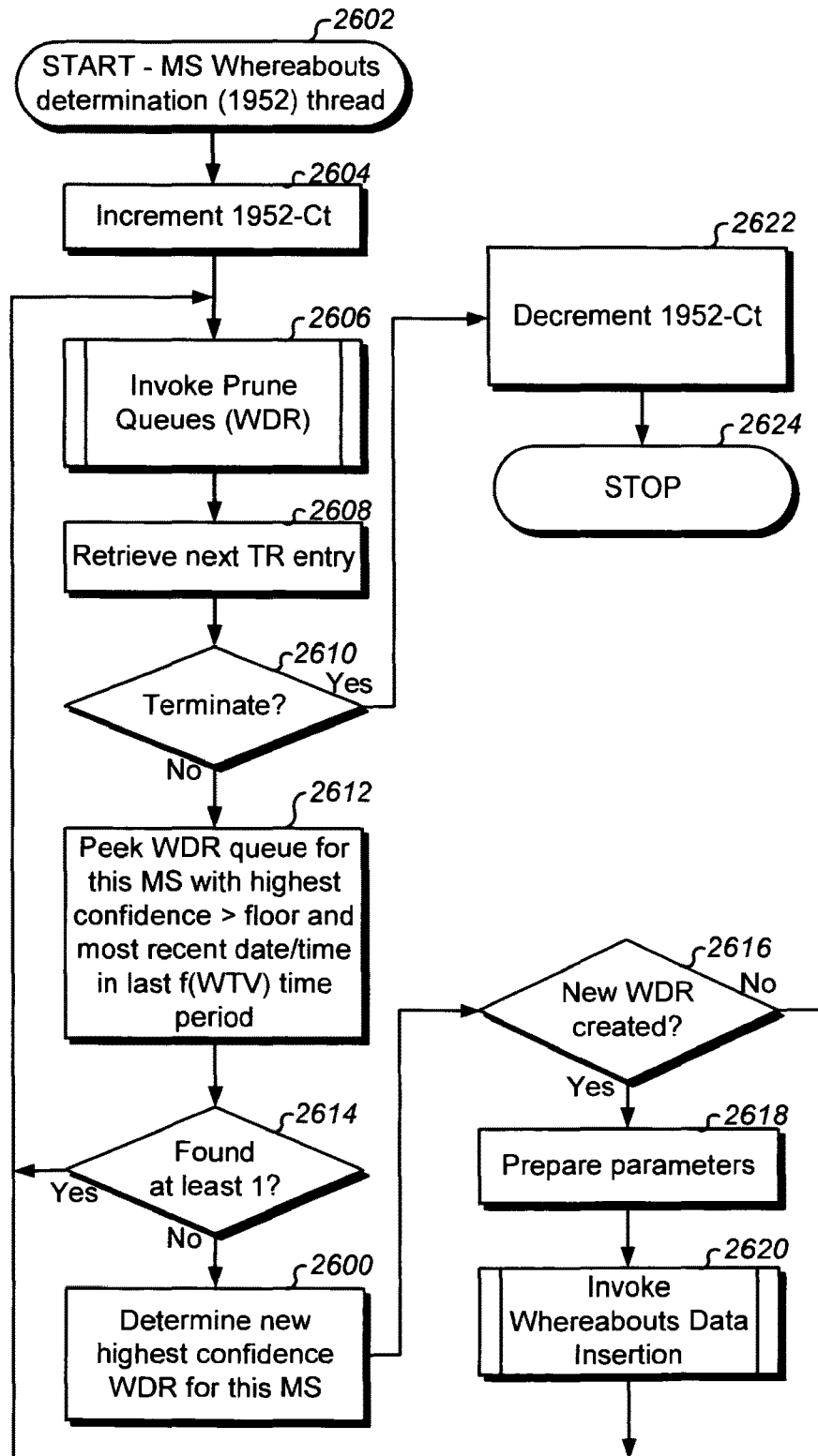
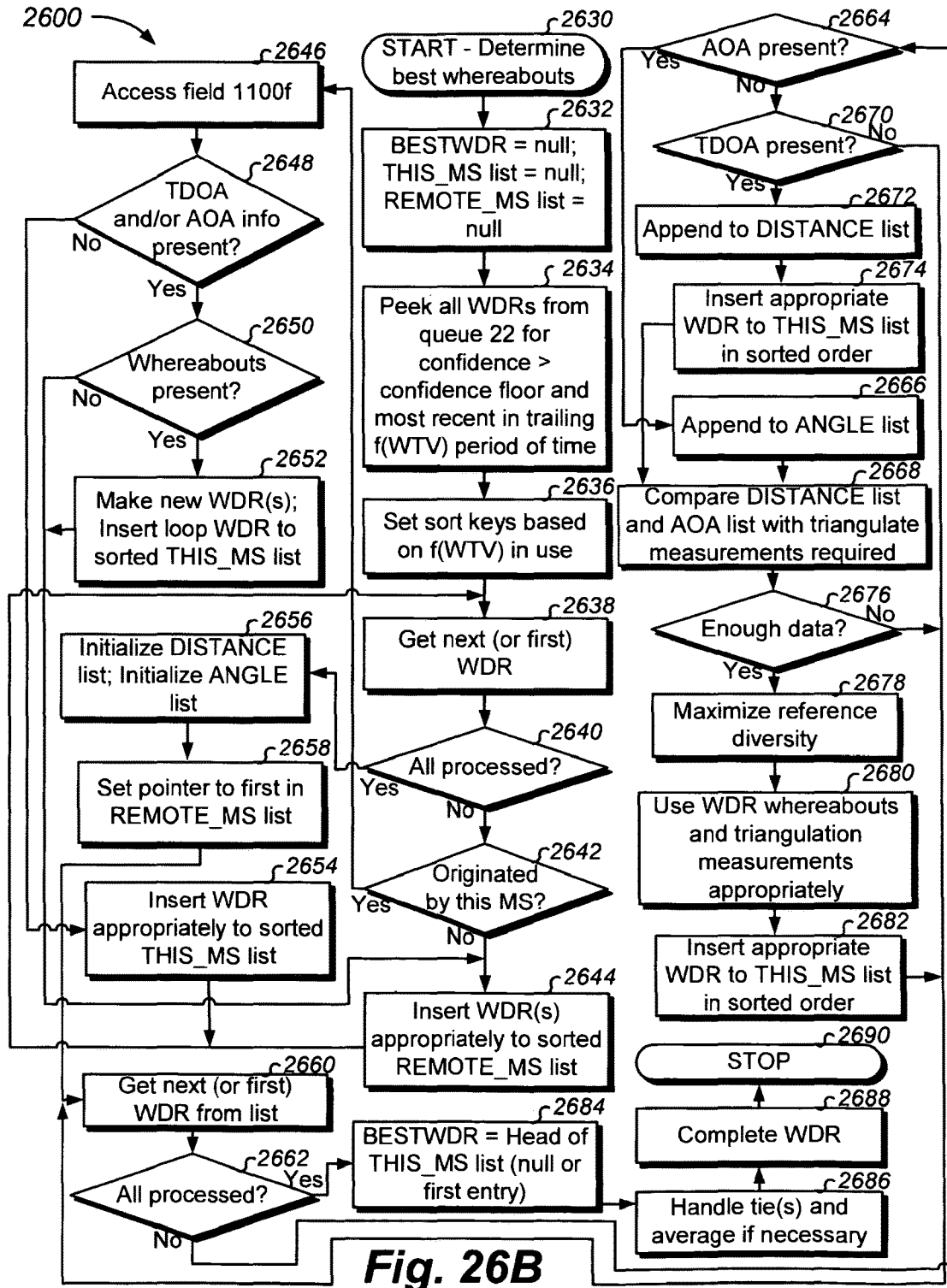
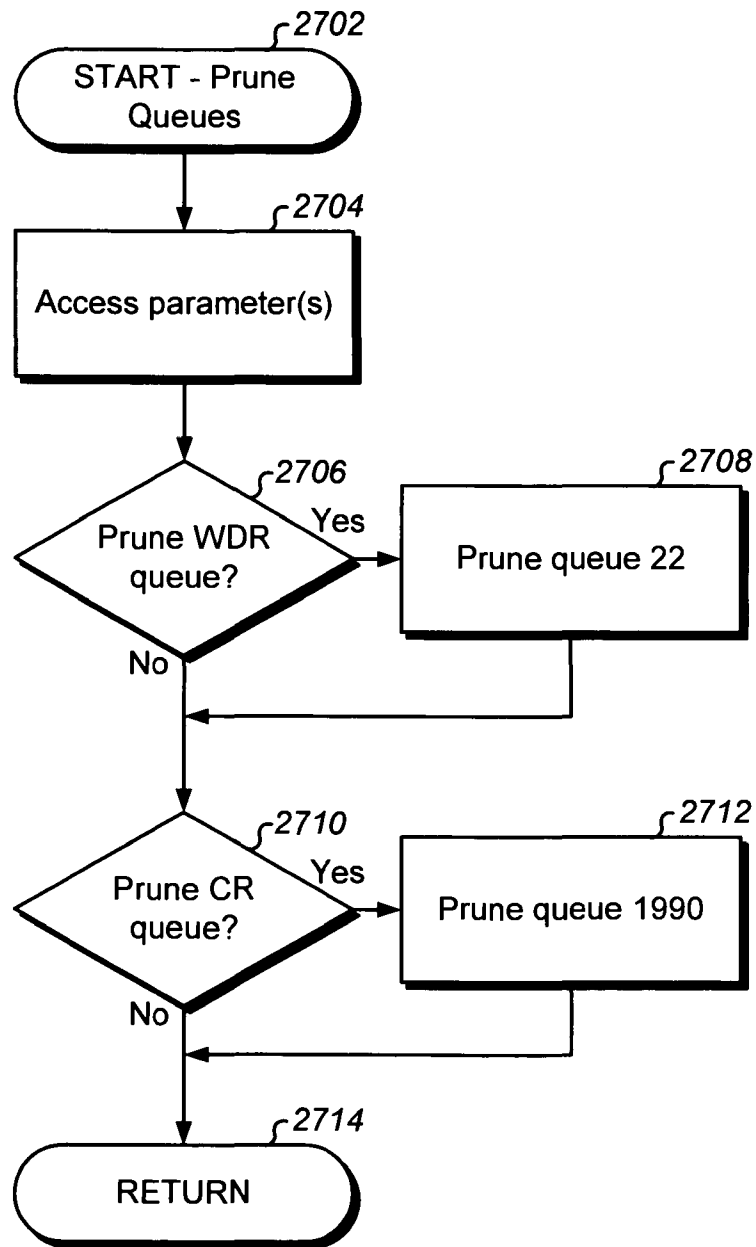


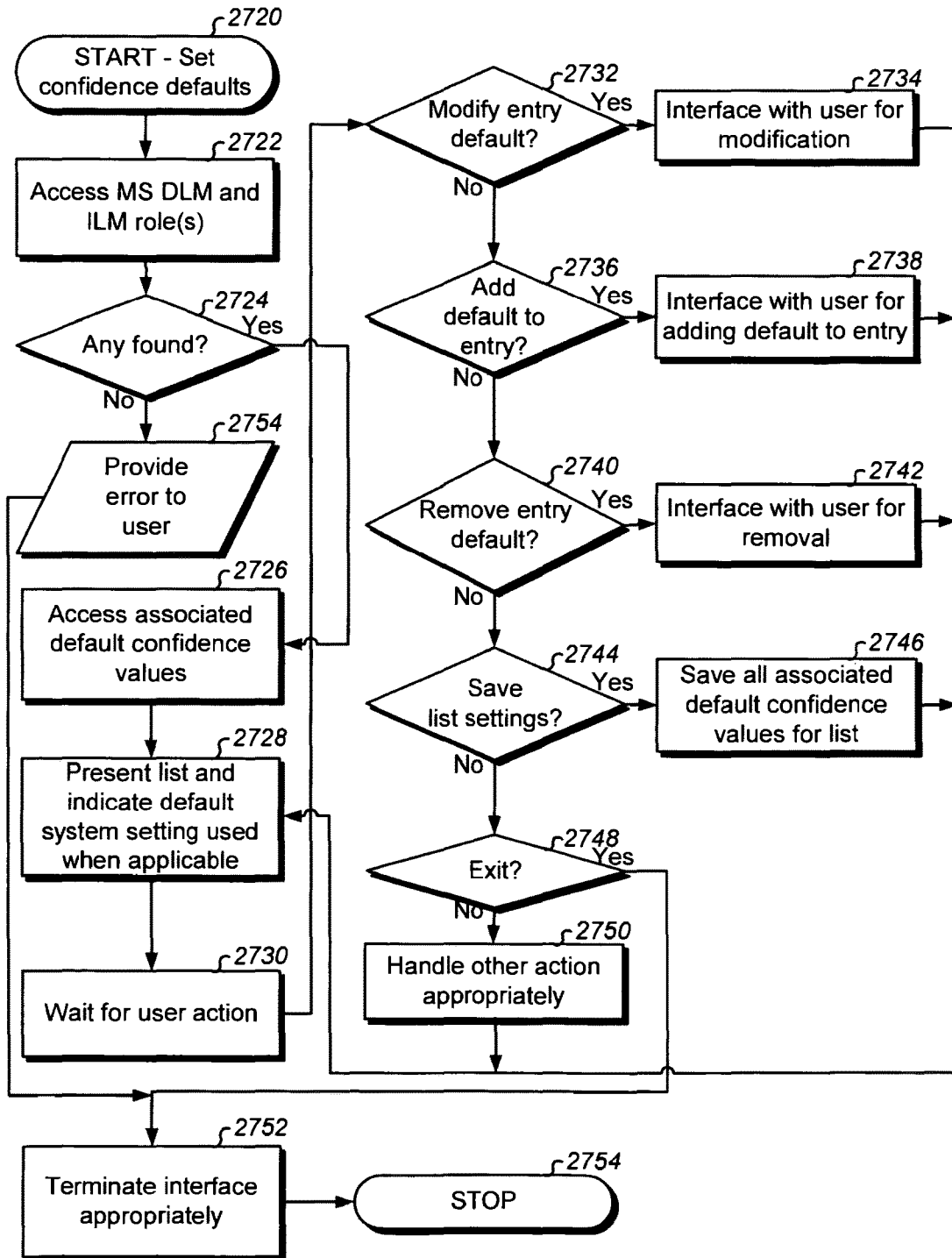
Fig. 24C

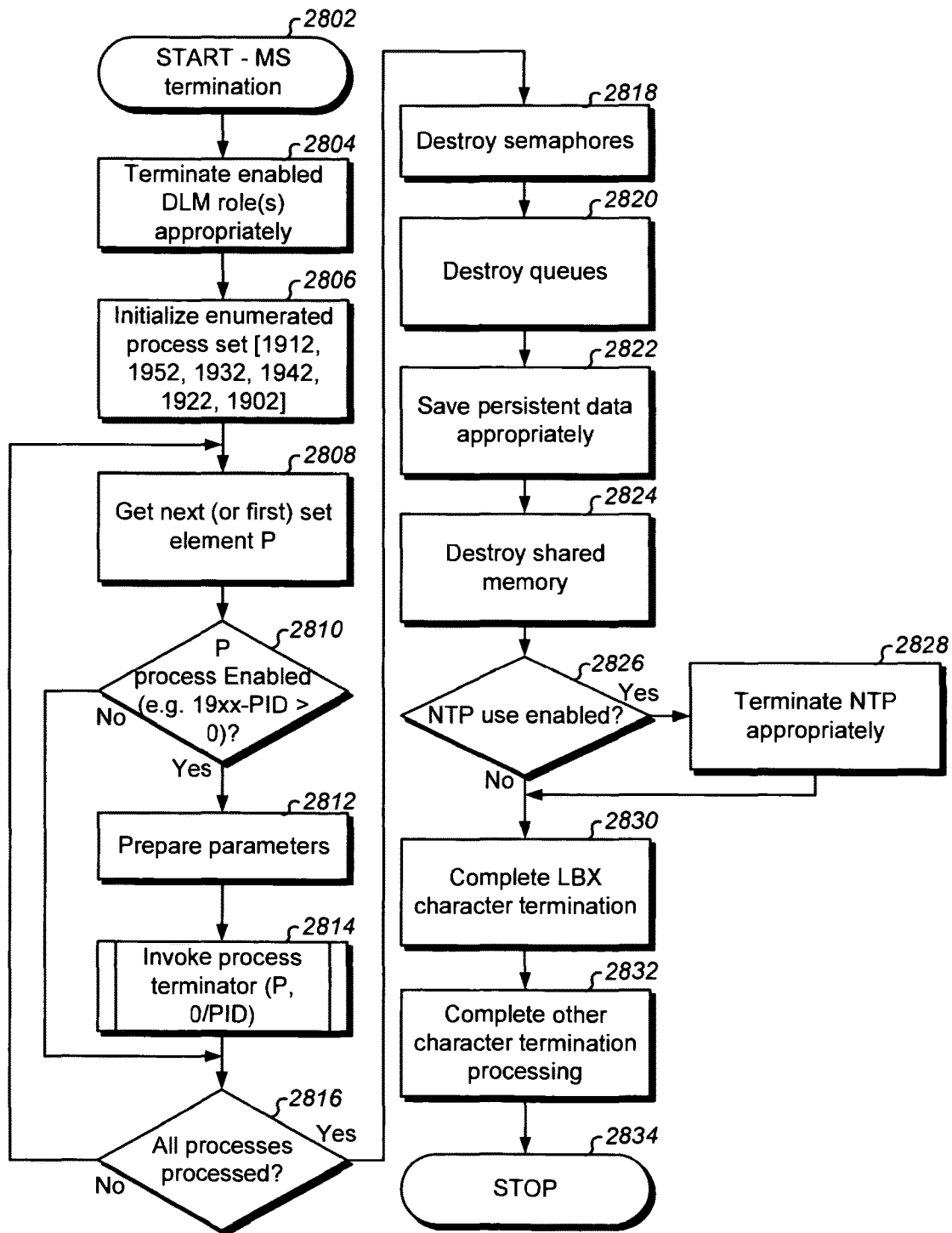
**Fig. 25**

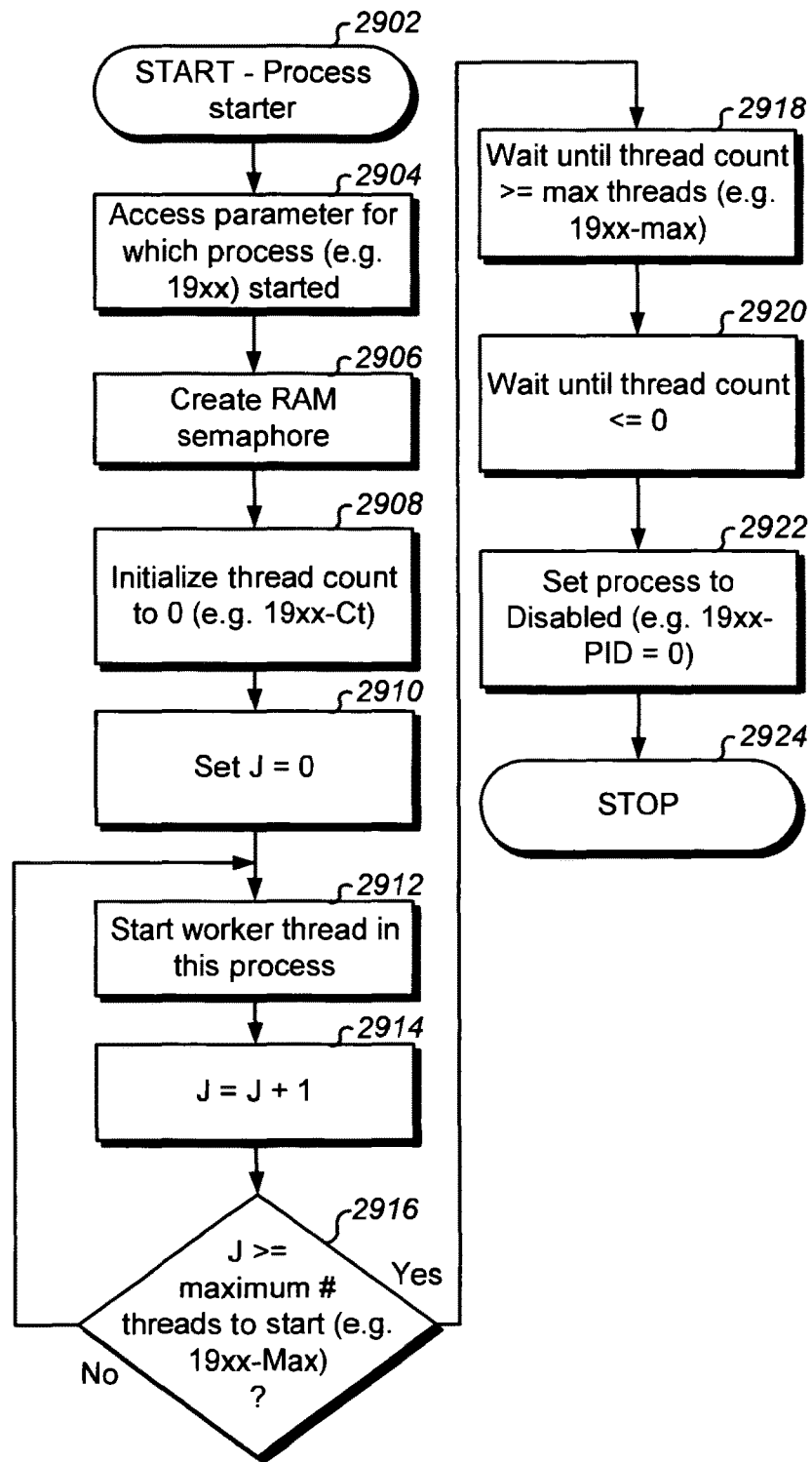
**Fig. 26A**

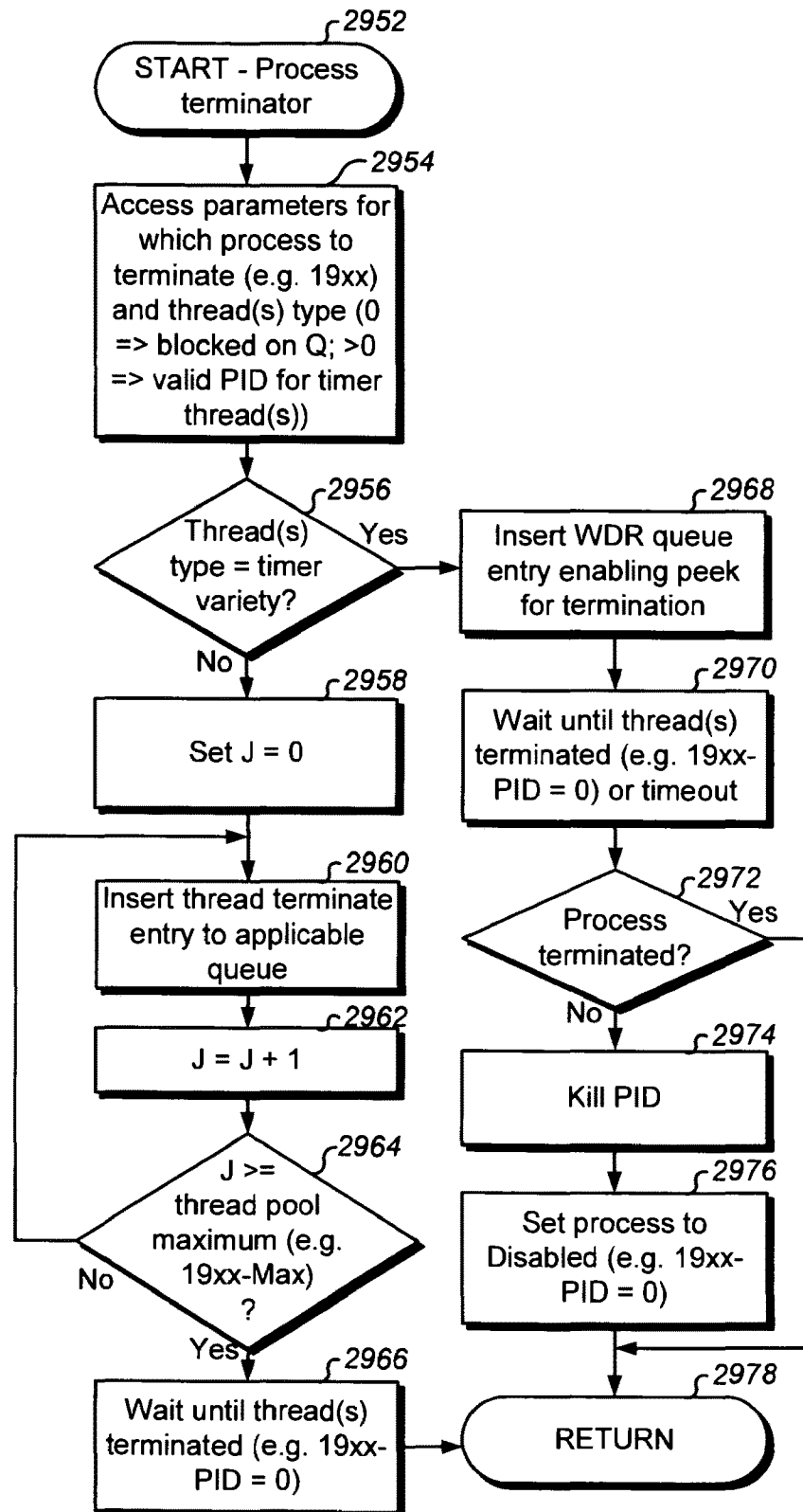


**Fig. 27A**

**Fig. 27B**

**Fig. 28**

**Fig. 29A**


**Fig. 29B**

U.S. Patent

Nov. 19, 2019

Sheet 73 of 322

US 10,477,994 B2

 3002a

// Figs. 30A through 30E syntaxes (e.g. delimiters, etc) used should enforce
 // appropriate unambiguous grammar parsability for Lex&Yacc, top down
 // recursive parsing, XML encoding, other syntactic embodiments, applicable semantic
 // representations, and any other syntactic/semantic embodiments. Figs. 30A through 30E BNF
 // grammar elaborates for a corresponding interpreter, recommended syntaxes, programming
 // language structures and/or objects, DB schemas, ANSI datastream encoding (e.g. X.409),
 // flowchart processing blocks and locations in parent application flowcharts, and any other
 // analogous implementation embodiments or subsets thereof.

// ***** Common BNF grammar (e.g. in Data 8): *****

Variables = "null" | Variables Variable
 // Variables are placed anywhere; Can be used for referencing (a="..." b=a c=b)

Variable = VarType(VarName) = "null" | VarType(VarName) = ...value(s)... |
 VarType(VarName) = [Variables] [VarInstantiations] |
 VarType(VarName) = [VarInstantiations] [Variables]
 // Variables scope to following & descending nesting; "value" has appropriate syntax
 // per VarType; VarName can be set to other variables (e.g. indirect tree structure)

VarInstantiations = "null" | VarInstantiations VarInstantiate

VarInstantiate = *VarName(Param1="x1", Param2="x2", ... ParamN="xN") for N >= 0
 // Parameters allow optionally substituting occurrences in VarName with new values
 // prior to instantiation:

VarName = "text string"

Description = "null" | "text string" | VarInstantiate

History = [CreatorInfo] [ModifierInfo] | VarInstantiations

CreatorInfo = "null" | [CreateDateTime] [CreatorID] [CreatorIDType]
 [CreatorAddr] [CreatorSysID] [CreatorSysType]
 [CreatorSysAddr] | VarInstantiations

ModifierInfo = "null" | [LastModifyDateTime] [LastModifyID]
 [LastModifyIDType] [LastModifyAddr] [LastModifySysID]
 [LastModifySysType] [LastModifySysAddr] | VarInstantiations

CreateDateTime = "date/time stamp" | VarInstantiate

CreatorID = ID

CreatorIDType = IDType

Fig. 30A

U.S. Patent

Nov. 19, 2019

Sheet 74 of 322

US 10,477,994 B2

3002b
↙


CreatorAddr = Address
CreatorSysID = "text string" | VarInstantiate
CreatorSysType = "system type" | VarInstantiate // e.g. type of MS
CreatorSysAddr = Address
LastModifyDateTime = "date/time stamp" | VarInstantiate
LastModifyID = ID
LastModifyIDType = IDType
LastModifyAddr = Address
LastModifySysID = "text string" | VarInstantiate
LastModifySysType = "system type" | VarInstantiate
LastModifySysAddr = Address
ID = "MS ID" [Description] [History] |
"MS Group ID" [Description] [History] | "User ID" [Description] [History] |
"User Group ID" [Description] [History] | "logical handle" [Description] [History] |
"physical handle" [Description] [History] | VarInstantiations
IDType = "MS_ID" | "MS_Group_ID" | "User_ID" | "User_Group_ID" |
"logical_handle" | "physical_handle" | VarInstantiate
Address = "ip address" | "SNA address" | "Postal address" |
"point" | "logical address" | "physical address" | "situational location" |
"2 dimensional area" | "3 dimensional area" | VarInstantiate
TimeSpec = "Xdate/time stamp" | "Xdate/time period" | VarInstantiate
VarType = Description | History | ID | IDType | CreatorInfo | ModifierInfo |
CreateDateTime | CreatorID | CreatorIDType | CreatorAddr | CreatorSysID |
CreatorSysType | CreatorSysAddr | LastModifyDateTime | LastModifyID |
LastModifyIDType | LastModifyAddr | LastModifySysID | LastModifySysType |
LastModifySysAddr | Address | "Xdate/time stamp" | "Xdate/time period" | "text string" |
"system type" | TimeSpec | "MS ID" | "MS Group ID" | "User ID" | "User Group ID" |
"logical handle" | physical handle" | "...Address elaborations..." |
"...IDType elaborations..." | Variable // | VarInstantiate here as well (but elaborates)

Fig. 30B

U.S. Patent

Nov. 19, 2019

Sheet 75 of 322

US 10,477,994 B2
 3034

// ***** BNF grammar for Permissions 10: *****

PermissionBody = "null" | [Variables] [Permissions]
 // [Variables] placed anywhere (not shown in constructs below to enhance readability)

Permissions = "null" | Permissions Permission | VarInstantiations

Permission = Grantor Grantee [Grants] [TimeSpec] [Description] [History] |
 VarInstantiations
 // No Grants implies granting all permissions; This embodiment ensures non-null
 // Grantor and Grantee, but "null" could be used (e.g. for placeholder entries).

Grantor = ID [IDType] | VarInstantiations
 // ID defaults (e.g. MS ID) when IDType not present

Grantee = ID [IDType] | VarInstantiations

Grants = "null" | Grants Grant | Privileges | VarInstantiations

Grant = "grant name" AND (Privileges [TimeSpec] [Description] [History] |
 Grants [TimeSpec] [Description] [History] |
 VarInstantiations)

Privileges = "null" | Privileges Privilege | VarInstantiations

Privilege = "atomic privilege for assignment" [MSRelevance]
 [TimeSpec] [Description] [History] | VarInstantiations

MSRelevance = "MS relevance descriptor"

Groups = "null" | Groups Group | VarInstantiations

Group = "group name" AND (IDs [Description] [History] |
 Groups [Description] [History] |
 VarInstantiations)

IDs = "null" | IDs ID [IDType] | VarInstantiations

VarType = *VarType | Permissions | Permission | Grantor | Grantee | Grants |
 Grant | Privileges | Privilege | MSRelevance | Groups | Group |
 IDs

Fig. 30C

U.S. Patent

Nov. 19, 2019

Sheet 76 of 322

US 10,477,994 B2

3068a
↙

```
// ***** BNF grammar for Charters 12: *****

CharterBody      = "null" | [ Variables ] [ Charters ]
                  // [ Variables ] placed anywhere (not shown in constructs below to enhance readability)

Charters          = "null" | Charters Charter | VarInstantiations

Charter           = Grantee Grantor Expression Actions [ TimeSpec ] [ Description ]
                  [ History ] | VarInstantiations

Expression        = Conditions [ TimeSpec ] | VarInstantiations
                  // This embodiment ensures at least one condition to a Charter, but "null" could be
                  // used (e.g. for placeholder entries).

Conditions         = Condition | Conditions CondOp Condition | VarInstantiations

CondOp            = "and" | "or" | VarInstantiations

Condition          = Term Op Term [ TimeSpec ] [ Description ] [ History ] |
                  Value [ TimeSpec ] [ Description ] [ History ] |
                  Invocation [ TimeSpec ] [ Description ] [ History ] | VarInstantiations
                  // Another embodiment allows unary operators (e.g. "not"), for example for boolean
                  // WDR fields (e.g. Applications field(s)). Current boolean tests for "True" or "False",
                  // or non-zero = "True" and zero = "False". Value & Invocation result in a boolean.

Term               = WDRTerm [ TimeSpec ] [ Description ] [ History ] |
                  AppTerm [ TimeSpec ] [ Description ] [ History ] |
                  Value [ TimeSpec ] [ Description ] [ History ] |
                  Invocation [ TimeSpec ] [ Description ] [ History ] |
                  PointSet [ TimeSpec ] [ Description ] [ History ] |
                  VarInstantiate

WDRTerm           = "Any WDR 1100 field, or any subset thereof" [ Description ]
                  [ History ] | VarInstantiate

AppTerm           = "Any Application data field, or any subset thereof" [ Description ]
                  [ History ] | VarInstantiate

Value             = Data | "number" | "text string" | "value" | "True" | "False" |
                  "atomic term" | "map term" | ID [ IDType ] | "null" | VarInstantiate

PointSet          = [ 2D | 3D ] [ Geo | Cartesian | Polar ]
                  "text string" [ Description ] [ History ] |
                  "numeric(s)" [ Description ] [ History ] |
                  Data [ Description ] [ History ] | VarInstantiate
```

Fig. 30D

3068b

```

Data                = "typed memory pointer" | "typed memory value" | "typed file path" |
                    "typed file path and offset" | "typed DB qualifier" | VarInstantiate
// i.e. pointer or value from stack, globals, shared memory, file data location, DB
// pointer, DB value, or any other data.

Invocation          = "DLL interface(optional params...)" |
                    "Linked interface(optional params...)" |
                    "executable path(optional params...)" | VarInstantiate
// Invocation can return any value of any type, except will be converted to a boolean
// when used as a Term (0 = False, else = True). Best to return boolean when Term use.

Op                  = [ "atomic not operator" ] "atomic operator" | ProfileMatch |
                    VarInstantiate

ProfileMatch        = "atomic profile match operator" | VarInstantiate

Actions             = "null" | Actions Action

Action              = [ Host ] Command Operand [Parameters]
                    [ TimeSpec ] [ Description ] [ History ] | VarInstantiations

Host                = "null" | ID [IDType] | VarInstantiations

Command             = "atomic command" | VarInstantiations
// Command may map to translation member entry of natural language map

Operand             = "atomic operand" | VarInstantiations
// Some embodiments have no need for an operand in this grammar (e.g. command file
// reference, DLL call, self contained command, invocation callout, etc).

Parameters          = "null" | Parameters Parameter | VarInstantiations

Parameter           = WDRTerm [ Description ] [ History ] |
                    AppTerm [ Description ] [ History ] |
                    Value [ Description ] [ History ] |
                    Invocation [ Description ] [ History ] |
                    ID [ IDType ] [ Description ] [ History ] |
                    VarInstantiate [ Description ] [ History ]

VarType             = *VarType | Charters | Charter | Expression | Conditions | Condition |
                    CondOp | WDRTerm | Term | Value | PointSet | Data | Invocation | Op |
                    Actions | ProfileMatch | Action | Command | Operand | Parameters |
                    Parameter | Host

```

Fig. 30E

Operand ↓	Command										
	101	103	105	119	107	109	111	113	115	117	...
201	#, sender, msg/subj, attribs, recip(s)	#, sender, msg/subj, attribs, recip(s)	#	#	#, system(s)	#, system(s)	#, ack, source, system(s)	#, ack, system(s)	#, ack, source, system(s)	#, system(s)	
203	link, sender, msg/subj, attribs, recip(s)	link, sender, msg/subj, attribs, recip(s)	link, params	link, params	link, params, system(s)	link, params, system(s)	link, ack, source, system(s)	link, ack, system(s)	link, ack, source, system(s)	link, params, system(s)	
205	body, sender, msg/subj, attribs, recip(s)	body, sender, msg/subj, attribs, recip(s)	body, sender, msg/subj, attribs, recip(s)	body, sender, msg/subj, attribs, recip(s)	email, system(s)	body, sender, msg/subj, attribs, recip(s)	email, ack, source, system(s)	email, ack, system(s)	email, ack, source, system(s)	email, system(s)	
207	msg, sender, msg/subj, attribs, recip(s)	msg, sender, msg/subj, attribs, recip(s)	msg, sender, msg/subj, attribs, recip(s)	body, sender, msg/subj, attribs, recip(s)	msg, system(s)	body, sender, msg/subj, attribs, recip(s)	msg, ack, source, system(s)	msg, ack, system(s)	msg, ack, source, system(s)	msg, system(s)	
209	body, sender, msg/subj, attribs, recip(s)	body, sender, msg/subj, attribs, recip(s)	body, sender, msg/subj, attribs, recip(s)	body, sender, msg/subj, attribs, recip(s)	email, system(s)	body, sender, msg/subj, attribs, recip(s)	email, ack, source, system(s)	email, ack, system(s)	email, ack, source, system(s)	email, system(s)	
211	msg, sender, msg/subj, attribs, recip(s)	msg, sender, msg/subj, attribs, recip(s)	msg, sender, msg/subj, attribs, recip(s)	msg, sender, msg/subj, attribs, recip(s)	msg, system(s)	body, sender, msg/subj, attribs, recip(s)	msg, ack, source, system(s)	msg, ack, system(s)	msg, ack, source, system(s)	msg, system(s)	
213	indicator, sender, msg/subj, attribs, recip(s)	indicator, sender, msg/subj, attribs, recip(s)	indicator, sender, msg/subj, attribs, recip(s)	indicator, sender, msg/subj, attribs, recip(s)	indicator, system(s)	indicator, system(s)	indicator, ack, source, system(s)	indicator, ack, system(s)	indicator, ack, source, system(s)	indicator, system(s)	

Fig. 31A

Operand ↓	Command										
	101	103	105	119	107	109	111	113	115	117	...
215	app, sender, msg/subj, attrs, recip(s)	app, sender, msg/subj, attrs, recip(s)	app, params	app, params	app, params, system(s)	app, params, system(s)	app, params, ack, source, system(s)	app, params, ack, system(s)	app, params, ack, source, system(s)	app, params, system(s)	
217	doc, sender, msg/subj, attrs, recip(s)	doc, sender, msg/subj, attrs, recip(s)	doc	doc	doc, system(s)	doc, system(s)	doc, ack, source, system(s)	doc, ack, system(s)	doc, ack, source, system(s)	doc, system(s)	
219	path, sender, msg/subj, attrs, recip(s)	path, sender, msg/subj, attrs, recip(s)	path	path	path, system(s)	path, system(s)	path, ack, source, system(s)	path, ack, system(s)	path, ack, source, system(s)	path, system(s)	
221	content, sender, msg/subj, attrs, recip(s)	content, sender, msg/subj, attrs, recip(s)	content	content	content, system(s)	content, system(s)	content, ack, source, system(s)	content, ack, system(s)	content, ack, source, system(s)	content, system(s)	
223	DB-obj, sender, msg/subj, attrs, recip(s)	DB-obj, query, sender, msg/subj, attrs, recip(s)	DB-obj	DB-obj, query	DB-obj, system(s)	DB-obj, query, system(s)	DB-obj, ack, source, system(s)	DB-obj, ack, system(s)	DB-obj, ack, source, system(s)	DB-obj, query, system(s)	
225	data, sender, msg/subj, attrs, recip(s)	data, value, sender, msg/subj, attrs, recip(s)	data, value	data, value	data, system(s)	data, value, system(s)	data, ack, source, system(s)	data, ack, system(s)	data, ack, source, system(s)	data, value, system(s)	

Fig. 31B

Operand ↓	Command										
	101	103	105	119	107	109	111	113	115	117	...
227	sem, sender, msg/subj, attrs, recip(s)	sem, cmd, sender, msg/subj, attrs, recip(s)	sem, cmd	sem, cmd	sem, system(s)	sem, cmd, system(s)	sem, ack, source, system(s)	sem, ack, system(s)	sem, ack, source, system(s)	sem, cmd, system(s)	
229	path, sender, msg/subj, attrs, recip(s)	path, sender, msg/subj, attrs, recip(s)	path	path	path, system(s)	path, system(s)	path, ack, source, system(s)	path, ack, system(s)	path, ack, source, system(s)	path, system(s)	
231	app, macro, sender, msg/subj, attrs, recip(s)	app, macro, sender, msg/subj, attrs, recip(s)	app, macro	app, macro	app, macro, system(s)	app, macro, system(s)	app, params, ack, source, system(s)	app, params, ack, system(s)	app, params, ack, source, system(s)	app, macro, system(s)	
233	"<alt> <prtscr>", sender, msg/subj, attrs, recip(s)	"<alt> <prtscr>", sender, msg/subj, attrs, recip(s)	"<alt> <prtscr>"	"<alt> <prtscr>"	objtxt, system(s)	cmds, system(s)	"<alt> <prtscr>", ack, source, system(s)	objtxt, ack, system(s)	"<alt> <prtscr>", ack, source, system(s)	"<alt> <prtscr>", system(s)	
235	macro, sender, msg/subj, attrs, recip(s)	macro, sender, msg/subj, attrs, recip(s)	macro	macro	macro, system(s)	macro, system(s)	macro, ack, system(s)	app, params, ack, system(s)	macro, ack, system(s)	macro, system(s)	
237	iodev, input, sender, msg/subj, attrs, recip(s)	iodev, input, sender, msg/subj, attrs, recip(s)	iodev, input	iodev, input	iodev, input, system(s)	iodev, input, system(s)	iodev, input, ack, system(s)	iodev, ack, system(s)	iodev, input, ack, system(s)	iodev, input, system(s)	

Fig. 31C

Operand ↓	Command										
	101	103	105	119	107	109	111	113	115	117	...
239	iodev, output, sender, subj, attrs, recip(s)	iodev, output, sender, subj, attrs, recip(s)	iodev, output	iodev, output	iodev, output, system(s)	iodev, output, system(s)	iodev, output, ack, system(s)	iodev, ack, system(s)	iodev, output, ack, system(s)	iodev, output, system(s)	
241	alert, sender, msg/subj, attrs, recip(s)	alert, sender, msg/subj, attrs, recip(s)	alert	alert	alert, system(s)	alert, system(s)	alert, ack, source, system(s)	alert, ack, system(s)	alert, ack, source, system(s)	alert, system(s)	
243	pid, signal, sender, msg/subj, attrs, recip(s)	pid, signal, sender, msg/subj, attrs, recip(s)	pid, signal	pid, signal	prname, system(s)	pid, signal, system(s)	prname, ack, source, system(s)	prname, ack, system(s)	prname, ack, source, system(s)	prname, signal system(s)	
245	container, sender, msg/subj, attrs, recip(s)	container, sender, msg/subj, attrs, recip(s)	container	container	container, system(s)	container, system(s)	container, ack, source, system(s)	container, ack, system(s)	container, ack, source, system(s)	container, system(s)	
247	progobj, data, sender, msg/subj, attrs, recip(s)	progobj, data, sender, msg/subj, attrs, recip(s)	progobj, data	progobj, data, sender, msg/subj, attrs, recip(s)	progobj, data, system(s)	progobj, data, system(s)	progobj, ack, source, system(s)	progobj, ack, system(s)	progobj, ack, source, system(s)	progobj, data, system(s)	
249	cursor, sender, msg/subj, attrs, recip(s)	cursor, sender, msg/subj, attrs, recip(s)	cursor	cursor, sender, msg/subj, attrs, recip(s)	cursor, system(s)	cursor, attrs, system(s)	ack, source, system(s)	ack, system(s)	ack, source, system(s)	cursor, attrs, system(s)	

Fig. 31D

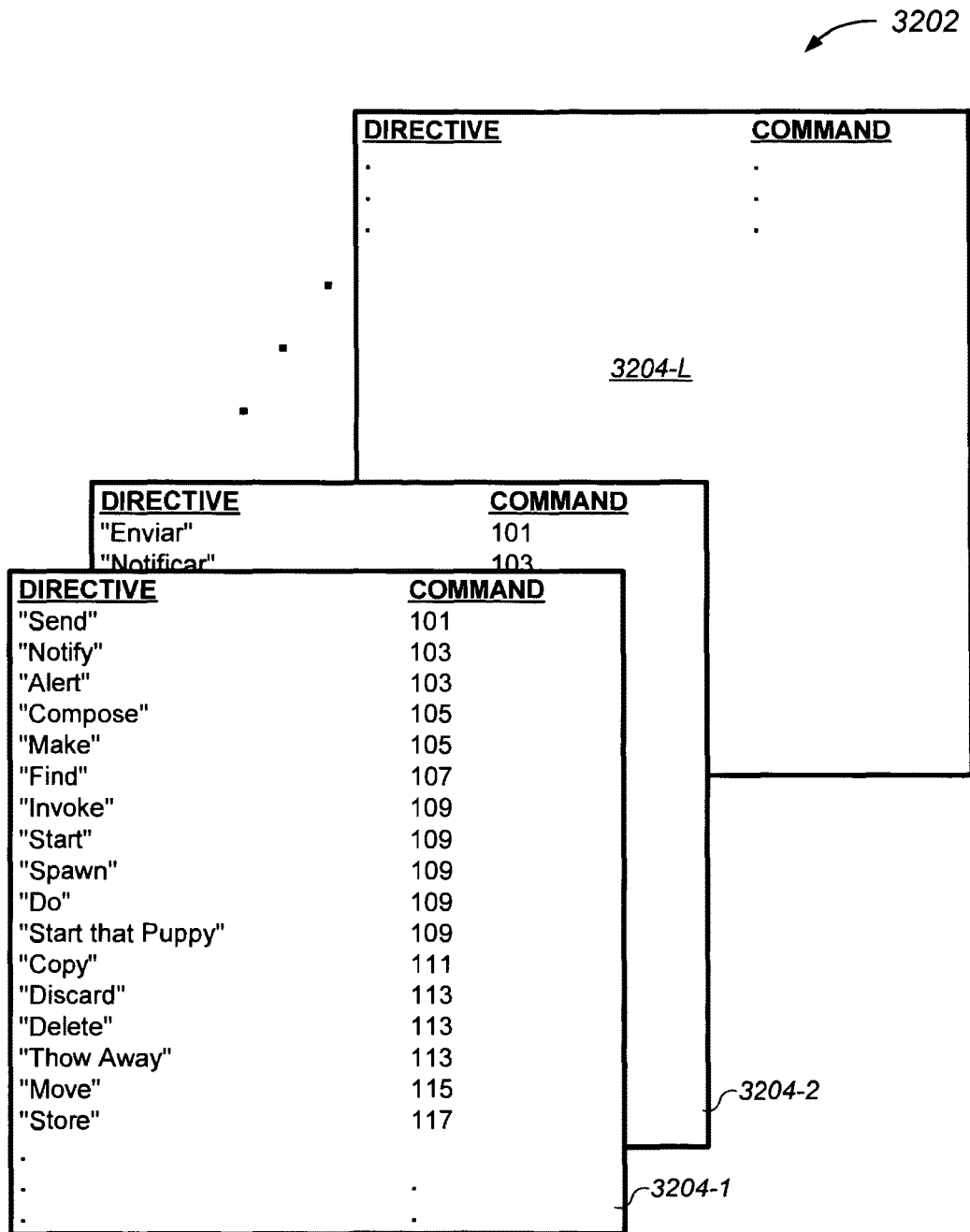
Operand ↓	Command										
	101	103	105	119	107	109	111	113	115	117	...
251	calobj, sender, msg/subj, attrs, recip(s)	calobj, sender, msg/subj, attrs, recip(s)	calobj, sender, msg/subj, attrs, recip(s)	calobj, sender, msg/subj, attrs, recip(s)	calobj, system(s)	calobj, attrs, system(s)	calobj, ack, source, system(s)	calobj, ack, system(s)	calobj, ack, source, system(s)	calobj, attrs, system(s)	
253	ABobj, sender, msg/subj, attrs, recip(s)	ABobj, sender, msg/subj, attrs, recip(s)	ABobj, sender, msg/subj, attrs, recip(s)	ABobj, sender, msg/subj, attrs, recip(s)	ABobj, system(s)	ABobj, attrs, system(s)	ABobj, ack, source, system(s)	ABobj, ack, system(s)	ABobj, ack, source, system(s)	ABobj, attrs, system(s)	
...											

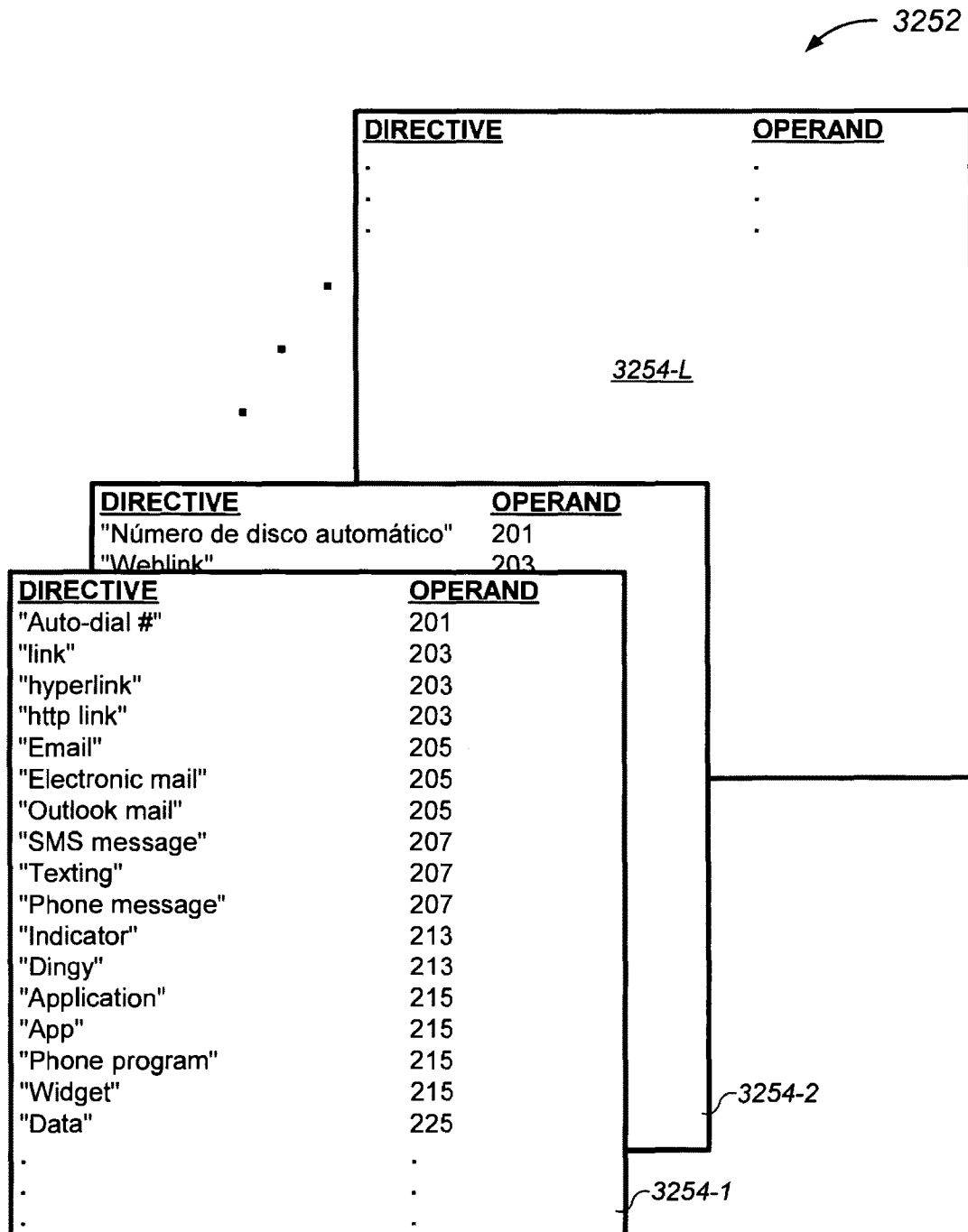
Fig. 31E

U.S. Patent

Nov. 19, 2019

Sheet 83 of 322

US 10,477,994 B2**Fig. 32A**

**Fig. 32B**

U.S. Patent

Nov. 19, 2019

Sheet 85 of 322

US 10,477,994 B2

<u>Token</u>	<u>Length</u>	<u>Value</u>
Variables ¹	L	complex ([Variable] ... [Variable]).
Variable	L	complex (First 2 bytes = VarType; VarName; value(s)); this can be present in any complex datastream for scope within current complex datastream thereafter and all descending constructs to it.
VarInstantiations ¹	L	complex ([VarInstantiate] ... [VarInstantiate]).
VarInstantiate	L	instantiation variable name and optional parameters; this can be subordinate to any other construct (e.g. {Description,8,{VarInstantiate,2,x}} where x is variable of string type (e.g. x = "Very long description text here"). Note the savings in TLV datastream size by using variables defined in 1 place for multiple subsequent instantiations thereafter).
VarName	L	text string for variable name.
Description	L	text string for description.
History	L	complex ([CreatorInfo] [ModifierInfo])
CreatorInfo	L	complex ([CreateDateTime] [CreatorID] [CreatorIDType] [CreatorAddr] [CreatorSysID] [CreatorSysType] [CreatorSysAddr]).
ModifierInfo	L	complex ([LastModifyDateTime] [LastModifyID] [LastModifyIDType] [LastModifyAddr] [LastModifySysID] [LastModifySysType] [LastModifySysAddr]).
CreateDateTime	L	date/time stamp (i.e. YYYYMMDDHHMMSS.12..J).
CreatorID	L	complex (ID).
CreatorIDType	1	Atomic element of IDType.
CreatorAddr	L	complex (Address).
CreatorSysID	L	Text string for system ID.
CreatorSysType	L	Text string for system type.
CreatorSysAddr	L	complex (Address).
LastModifyDateTime	L	date/time stamp.
LastModifyID	L	complex (ID).
LastModifyIDType	L	Atomic element of IDType.
LastModifyAddr	L	complex (Address).
LastModifySysID	L	Text string for system ID.
LastModifySysType	L	Text string for system type.
LastModifySysAddr	L	complex (Address).
ID	L	complex (first 2 bytes = length of the identifier; followed by identifier;[Description] [History]).
IDType	1	Atomic element of IDType.
Address	L	First 2 bytes = address type; next L-2 bytes = address info.
TimeSpec	L	First byte = spec type (stamp, period); Next bytes = the time spec(s) (e.g. preferably in syntax described).

Fig. 33A

U.S. Patent**Nov. 19, 2019****Sheet 86 of 322****US 10,477,994 B2**

<u>Token</u>	<u>Length</u>	<u>Value</u>
PermissionBody	L	complex ([Variables] [Permissions]).
Permissions ¹	L	complex ([Permission] .. [Permission]).
Permission	L	complex (Grantor Grantee [Grants] [TimeSpec] [Description] [History]).
Grantor	L	complex (ID [IDType]).
Grantee	L	complex (ID [IDType]).
Grants ¹	L	complex ([Grant] ... [Grant] [Privileges]).
Grant	L	First 2 bytes = length of Grant name; following bytes = grant name string; then complex: (Privileges [TimeSpec] [Description] [History] Grants [TimeSpec] [Description] [History]).
Privileges ¹	L	complex ([Privilege] ... [Privilege]).
Privilege	L	first 4 bytes = unsigned integer for atomic privilege assigned; following bytes (L-4) are complex: ([MSRelevance] [TimeSpec] [Description] [History]).
MSRelevance	8	64 bits for up to 64 different MS types of different capabilities.
Groups ¹	L	complex ([Group] ... [Group]).
Group	L	First 2 bytes = length of Group name; following bytes = group name string; then complex: (IDs [Description] [History] Groups [Description] [History]).
IDs ¹	L	complex (ID [IDType] ... ID [IDType]).

Fig. 33B

U.S. Patent

Nov. 19, 2019

Sheet 87 of 322

US 10,477,994 B2

<u>Token</u>	<u>Length</u>	<u>Value</u>
CharterBody	L	complex ([Variables] [Charters]).
Charters ¹	L	complex ([Charter] ... [Charter]).
Charter	L	complex (Grantee Grantor Expression Actions [TimeSpec] [Description] [History]).
Expression	L	complex (Conditions [TimeSpec]).
Conditions	L	complex (Condition Conditions CondOp Condition).
CondOp	1	"&" or " ".
Condition	L	complex (Term Op Term [TimeSpec] [Description] [History] Value [TimeSpec] [Description] [History] Invocation [TimeSpec] [Description] [History]).
Term	L	complex (WDRTerm [TimeSpec] [Description] [History] AppTerm [TimeSpec] [Description] [History] Value [TimeSpec] [Description] [History] Invocation [TimeSpec] [Description] [History]).
WDRTerm	L	first 2 bytes for WDR 1100 field/subfield length; following bytes are __name syntactical reference; then, if any, is complex = [Description] [History]).
AppTerm	L	first 2 bytes for Application field length; following bytes are Prefix_name syntactical reference; then, if any, is complex = [Description] [History]).
Value	L	first byte indicates Value type; following bytes (L-1), if any, is the "number", "text string", "value", "Boolean", "null", "atomic term", "map term" or complex = (Data ID [IDType]).
PointSet	L	first byte indicates dimension; second byte indicates type; third byte indicates format; next 2 bytes is the point set information length (LEN); following bytes is the PointSet information (may be complex for Data), following bytes (L-5-LEN), if any, is complex = [Description] [History]).
Data	L	first byte = atomic element data type; L-1 following bytes are the data syntactical reference.
Invocation	L	first byte = atomic element data type; L-1 following bytes = atomic element invocation with optional parameters.
Op	2	the operator reference (not clarifier simply provides unique operator (e.g. = and != are two operators; ProfileMatch here too). Numeric values used instead of characters here.

Fig. 33C-1

U.S. Patent

Nov. 19, 2019

Sheet 88 of 322

US 10,477,994 B2

<u>Token</u>	<u>Length</u>	<u>Value</u>
Actions ¹	L	complex ([Action] ... [Action]).
Action	L	complex ([Host] Command Operand [Parameters] [TimeSpec] [Description] [History]).
Host	L	complex (ID [IDType]).
Command	2	the command reference.
Operand	2	the operand reference.
Parameters ¹	L	complex ([Parameter] ... [Parameter]).
Parameter	L	complex (WDRTerm [Description] [History] AppTerm [Description] [History] Value [Description] [History] Invocation [Description] [History] ID [IDType] [Description] [History]).

Fig. 33C-2

```

//***** Grammar Common Definitions: *****
//
#define    TOKEN_LENGTH                2
#define    LENGTH_LENGTH              4

// #define    VARTYPE_x                Use Token Definitions for VarType

#define    IDTYPE_MSID                 11
#define    IDTYPE_MSGRPID              12
#define    IDTYPE_USERID               13
#define    IDTYPE_USERGRPID            14
#define    IDTYPE_LOGICAL              15
// e.g. ip address and socket; e.g. inetd.cfg invocation (e.g. 23.56.232.2:34002)
#define    IDTYPE_PHYSICAL              16 // MS serial #

#define    ADDRTYPE_LOGICAL            21 // e.g. ip address
#define    ADDRTYPE_PHYSICAL           22 // e.g. MS serial #
#define    ADDRTYPE_POSTAL             23
#define    ADDRTYPE_POINT              24
#define    ADDRTYPE_SL                 25
#define    ADDRTYPE_2D                 26
#define    ADDRTYPE_3D                 27

#define    TIMESPECTYPE_STAMP           31
#define    TIMESPECTYPE_PERIOD         32

```

Fig. 34A

U.S. Patent

Nov. 19, 2019

Sheet 90 of 322

US 10,477,994 B2

//***** Grammar Common Construct Token Definitions: *****

//

// #define	VARIABLES	10001
#define	VARIABLE	10002
// #define	VARINstantiations	10003
#define	VARINstantiate	10004
#define	VARNAME	10005
#define	DESCRIPTION	10006
#define	HISTORY	10007
#define	CREATORINFO	10008
#define	MODIFIERINFO	10009
#define	CREATEDATETIME	10010
#define	CREATORID	10011
#define	CREATORIDTYPE	10012
#define	CREATORADDR	10013
#define	CREATORSYSID	10014
#define	CREATORSYSTYPE	10015
#define	CREATORSYSADDR	10016
#define	LASTMODIFYDATETIME	10017
#define	LASTMODIFYID	10018
#define	LASTMODIFYIDTYPE	10019
#define	LASTMODIFYADDR	10020
#define	LASTMODIFYSYSID	10021
#define	LASTMODIFYSYSTYPE	10022
#define	LASTMODIFYSYSADDR	10023
#define	ID	10024
#define	IDTYPE	10025
#define	ADDRESS	10026
#define	TIMESPEC	10027

//***** Grammar Permission Construct Token Definitions: *****

//

#define	PERMISSIONBODY	12001
// #define	PERMISSIONS	12002
#define	PERMISSION	12003
#define	GRANTOR	12004
#define	GRANTEE	12005
#define	GRANTS	12006
#define	GRANT	12007
// #define	PRIVILEGES	12008
#define	PRIVILEGE	12009
#define	MSRELEVANCE	12010

Fig. 34B


```
#define GROUPS 12011
#define GROUP 12012
#define IDS 12013
```

```
//***** Grammar Charter Construct Token Definitions: *****
```

```
//
```

```
#define CHARTERBODY 14001
// #define CHARTERS 14002
#define CHARTER 14003
#define EXPRESSION 14004
#define CONDITIONS 14005
#define CONDOP 14006
#define CONDITION 14007
#define TERM 14008
#define WDRTERM 14009
#define APPTERM 14010
#define VALUE 14011
#define DATA 14012
#define INVOCATION 14013
#define OP 14014
// #define ACTIONS 14015
#define ACTION 14016
#define HOST 14017
#define COMMAND 14018
#define OPERAND 14019
// #define PARAMETERS 14020
#define PARAMETER 14021
#define POINTSET 14022
```

```
//***** Grammar Charter Definitions: *****
```

```
//
```

```
// atomic terms (e.g. \loc_my), WDR field terms (e.g. __location),
// Application terms (e.g. M_source), Invocation (e.g. fcn(p1,p2)), CondOp (e.g. "&") and
// Data atomic elements (e.g. c:\dir1\fname:58/LONGINT) are recognized syntaxes.
```

```
#define VALUE_NUMBER 41
#define VALUE_TEXT 42
#define VALUE_ENUM 43 // "value"
#define VALUE_BOOLEAN 44 // 1 = True, 0 = False
#define VALUE_ID 45
```

```
#define DIMENSION2 71
#define DIMENSION3 72
#define FORMAT_GEO 73
#define FORMAT_CARTESIAN 74
#define FORMAT_POLAR 75
```

Fig. 34C

```

//***** Atomic Commands : *****
//
#define      CMD_SEND      101
#define      CMD_NOTIFY    103
#define      CMD_COMPOSE   105
#define      CMD_FIND      107
#define      CMD_INVOKE    109
#define      CMD_COPY       111
#define      CMD_DISCARD   113
#define      CMD_MOVE      115
#define      CMD_STORE      117
#define      CMD_CONNECT   119
#define      CMD_ADMINISTRATE 121
#define      CMD_CHANGE    123

//***** Atomic Operands : *****
//
#define      OPERAND_AUTODIALNUMBER 201
#define      OPERAND_WEBLINK        203
#define      OPERAND_EMAIL          205
#define      OPERAND_SMSMSG         207
#define      OPERAND_BRDEMAIL       209
#define      OPERAND_BRDSMSMSG      211
#define      OPERAND_INDICATOR      213
#define      OPERAND_APP            215
#define      OPERAND_DOCUMENT       217
#define      OPERAND_FILE           219
#define      OPERAND_CONTENT        221
#define      OPERAND_DBOBJ          223
#define      OPERAND_DATA           225
#define      OPERAND_SEMAPHORE      227
#define      OPERAND_DIRECTORY      229
#define      OPERAND_APPCONTEXT     231
#define      OPERAND_UIFOBJ         233
#define      OPERAND_UIFCTL         235
#define      OPERAND_INPUT          237
#define      OPERAND_OUTPUT         239
#define      OPERAND_ALERT          241
#define      OPERAND_PROC           243
#define      OPERAND_CONTAINER      245
#define      OPERAND_PROGOBJ        247
#define      OPERAND_CURSOR         249
#define      OPERAND_CALENDAR       251
#define      OPERAND_ADDRESSBOOK    253

```

Fig. 34D

```

// TIMESPEC date/time stamps for open ended periods are set with no start/end spec:
// >=YYMMDDHHMMSS.1..J ==> set x.endDT to DT_NOENDSPEC;
// <=YYMMDDHHMMSS.1..J ==> set x.startDT to DT_NOSTARTSPEC;
// <YYMMDDHHMMSS.1..J and >YYMMDDHHMMSS.1..J subtracts/adds min precision
// from specified date/time stamp (i.e. TIMESPEC periods are preferably inclusive).
#define DT_NOENDSPEC          -1.0
#define DT_NOSTARTSPEC        -2.0

typedef struct timespec { // specifications converted to a Julian period form
    double          startDT;    // converted to Julian format (1ms precision)
    double          endDT;      // converted to Julian format (1ms precision)
    struct timespec *nextTS;    // linked list of sibling timespecs
} TIMESPEC;

typedef struct {
    double          *dt;        // Julian date/time
    unsigned char   id[MAX_IDLENGTH];
    unsigned short  idtype;     // for IDTYPE_x values
    // unsigned short cadr_type; // Assume 1 format here
    unsigned char   *c_address;
    char            *sysid;
    char            *systype;
    // unsigned short sysadr_type; // Assume 1 format here
    unsigned char   *sys_address;
} BOOKKEEP;

typedef struct {
    BOOKKEEP *creation;
    BOOKKEEP *modify;
} HISTORY;

typedef struct {
    unsigned short  vartype;
    char            name[MAX_VARNAME];
    unsigned char   *value;     // may be complex or series of complex
} VAR;

```

Fig. 34E

```

typedef struct privilege {
    unsigned long    priv;           // constant value of known privilege
    unsigned char    relevance[MAX_RELEVANCEMASK];
    TIMESPEC        *tspec;
    char            *desc;
    HISTORY         *hist;
    struct privilege *nextPriv;     // linked list of sibling privileges
} PRIVILEGE;

typedef struct grant {
    char            name[MAX_GRNAMLENGTH];
    char            permttype;      // 'P' = Privilege(s), 'G' = Grant(s)
    union {
        struct grant *grants;      // linked list subordinate/descending grant(s)
        PRIVILEGE    *privileges;  // linked list of privilege(s)
    } assigned;
    TIMESPEC        *tspec;
    char            *desc;
    HISTORY         *hist;
    struct grant    *nextGrant;    // linked list of sibling grants
} GRANT;

typedef struct permission {
    unsigned char    grantor[MAX_IDLENGTH];
    unsigned short   gortype;       // for IDTYPE_x values
    unsigned char    grantee[MAX_IDLENGTH];
    unsigned short   geetype;       // for IDTYPE_x values
    char            permttype;      // 'P' = Privilege(s), 'G' = Grant(s)
    union {
        GRANT        *grants;      // linked list of grant(s)
        PRIVILEGE    *privileges;  // linked list of privilege(s)
    } assigned;
    TIMESPEC        *tspec;
    char            *desc;
    HISTORY         *hist;
    struct permission *nextPerm;    // linked list of permissions
} PERMISSION;

```

Fig. 34F

```

typedef struct identity {
    unsigned char    id[MAX_IDLENGTH];
    unsigned short   idtype;        // for IDTYPE_x values
    struct identity  *nextID;       // linked list of sibling IDs
} IDENTITY;

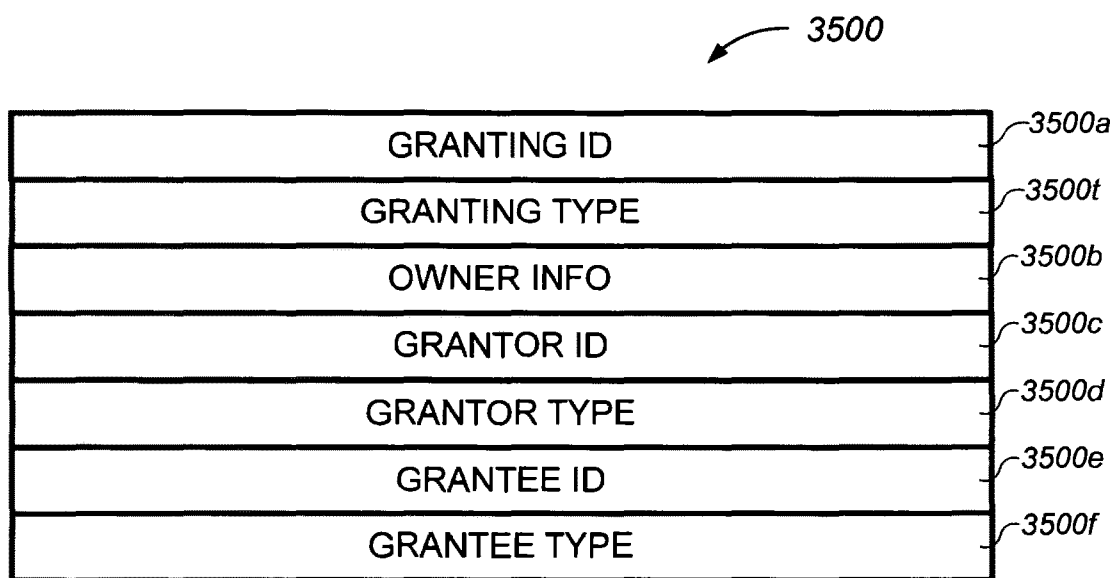
typedef struct group {
    char             name[MAX_GRPNAMELENGTH];
    char             grptype;       // 'B' = Branch, 'L' = Leaf
    union {
        struct group *groups;       // linked list subordinate/descending group(s)
        IDENTITY     *ids;          // linked list of IDs in this group
    } assigned;
    char             *desc;
    HISTRY           *hist;
    struct grant      *nextGroup;   // linked list of sibling groups
} GROUP;

typedef struct action {
    IDENTITY          host;         // .idtype = 0 = no host spec)
    unsigned short    cmd;
    unsigned short    operand;
    unsigned char     *params;      // maintained in syntax for flexibility
                                   // and for stack processing
    TIMESPEC         *tspec;
    char             *desc;
    HISTRY           *hist;
    struct action     *nextActn;    // linked list of sibling actions
} ACTION;

typedef struct charter {
    unsigned char     grantee[MAX_IDLENGTH];
    unsigned short    geetype;      // for IDTYPE_x values
    unsigned char     grantor[MAX_IDLENGTH];
    unsigned short    gortype;      // for IDTYPE_x values
    TIMESPEC         *exprTS;
    unsigned char     *cond;        // at least 1 condition maintained in
                                   // syntax for proper stack processing
    ACTION           *actn;
    char             *desc;
    HISTRY           *hist;
    struct charter    *nextCharter; // linked list of charters
} CHARTER;

```

Fig. 34G

**Fig. 35A**

U.S. Patent

Nov. 19, 2019

Sheet 97 of 322

US 10,477,994 B2

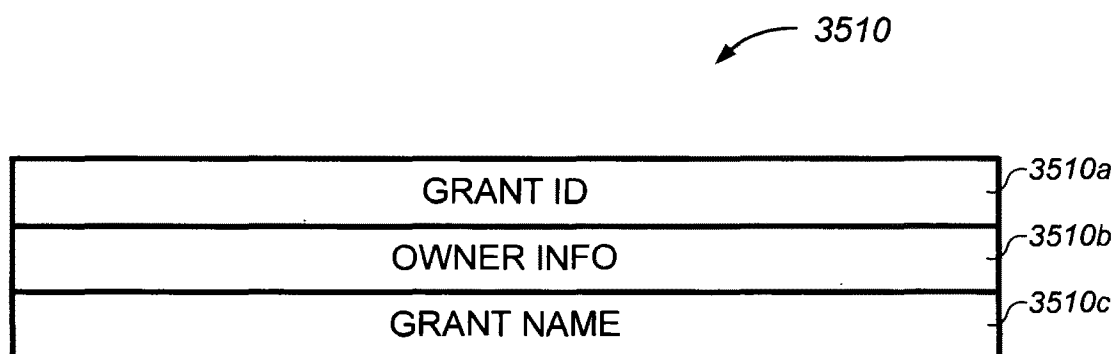


Fig. 35B

U.S. Patent

Nov. 19, 2019

Sheet 98 of 322

US 10,477,994 B2

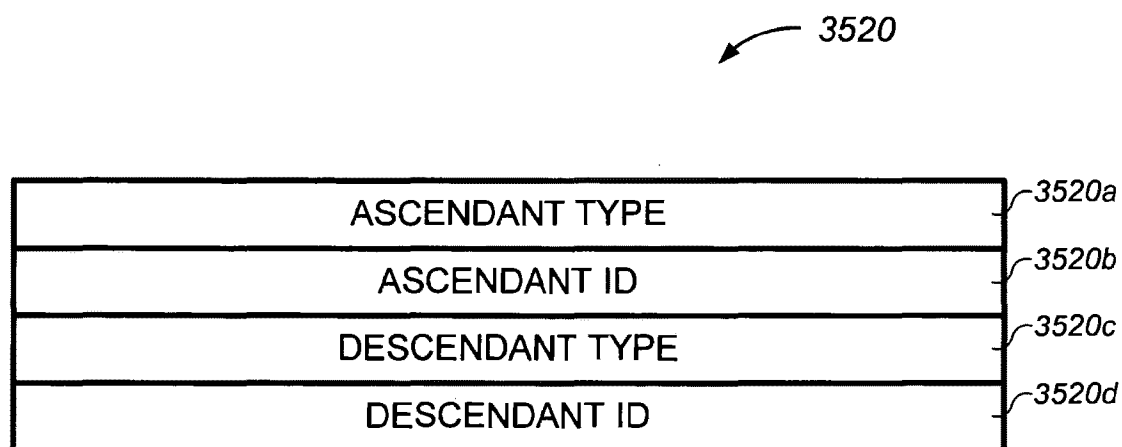


Fig. 35C

U.S. Patent

Nov. 19, 2019

Sheet 99 of 322

US 10,477,994 B2

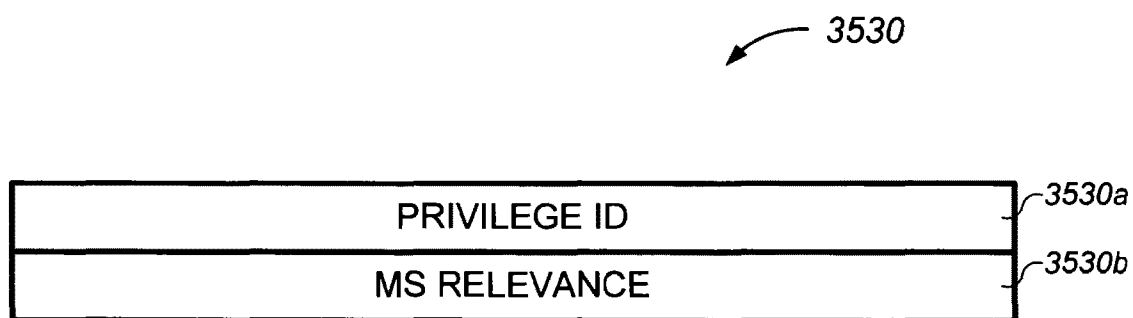


Fig. 35D

U.S. Patent

Nov. 19, 2019

Sheet 100 of 322

US 10,477,994 B2

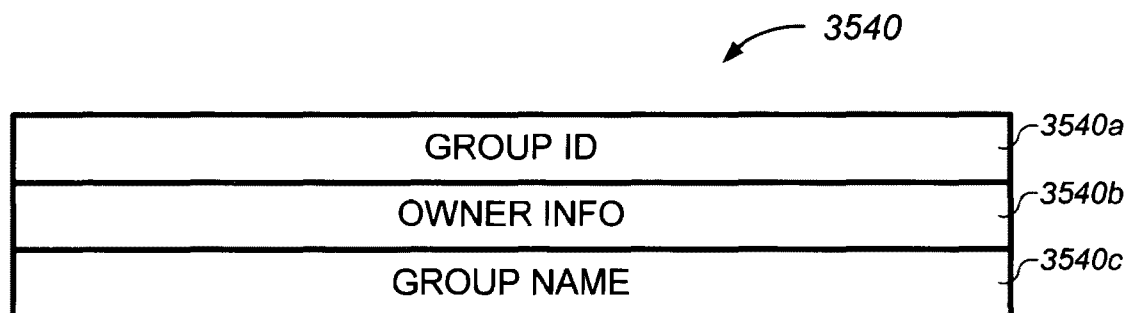


Fig. 35E

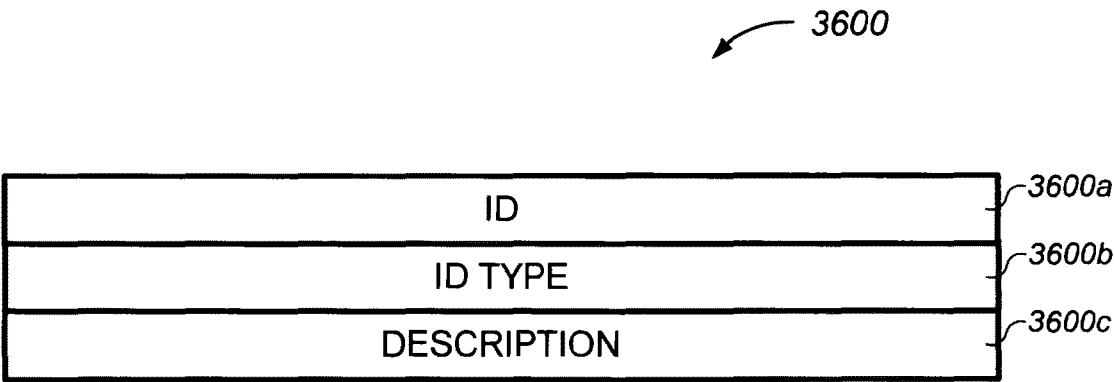


Fig. 36A

U.S. Patent

Nov. 19, 2019

Sheet 102 of 322

US 10,477,994 B2

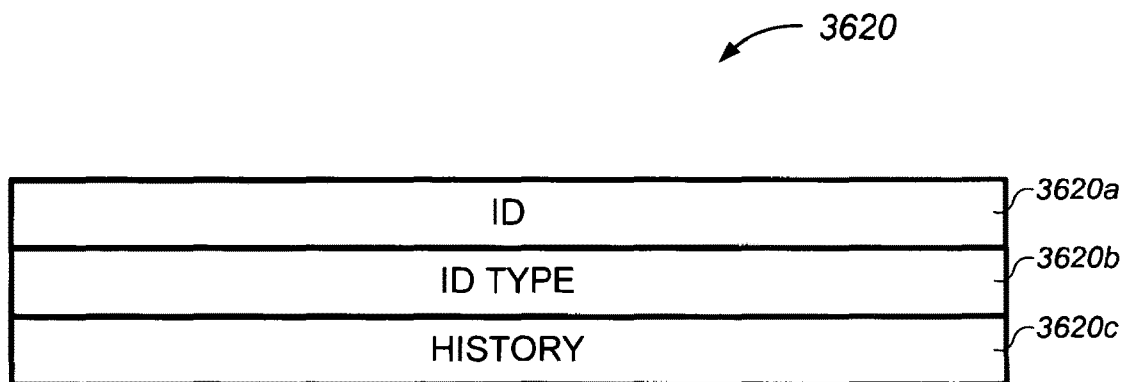


Fig. 36B

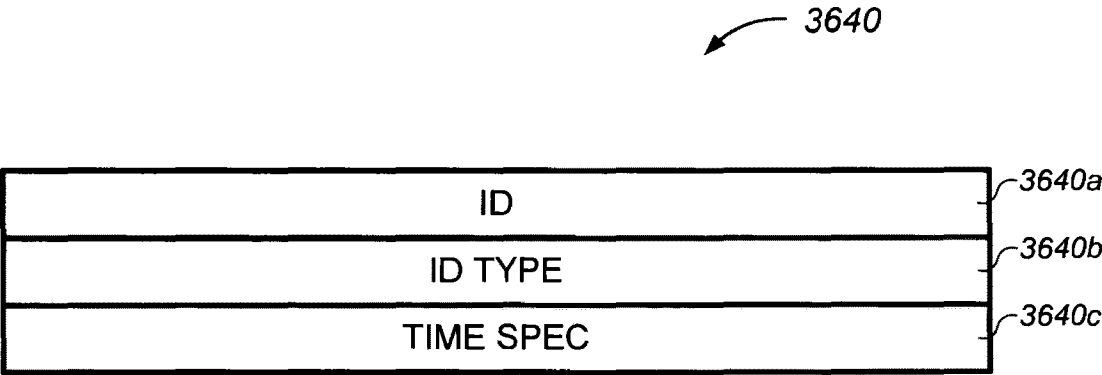


Fig. 36C

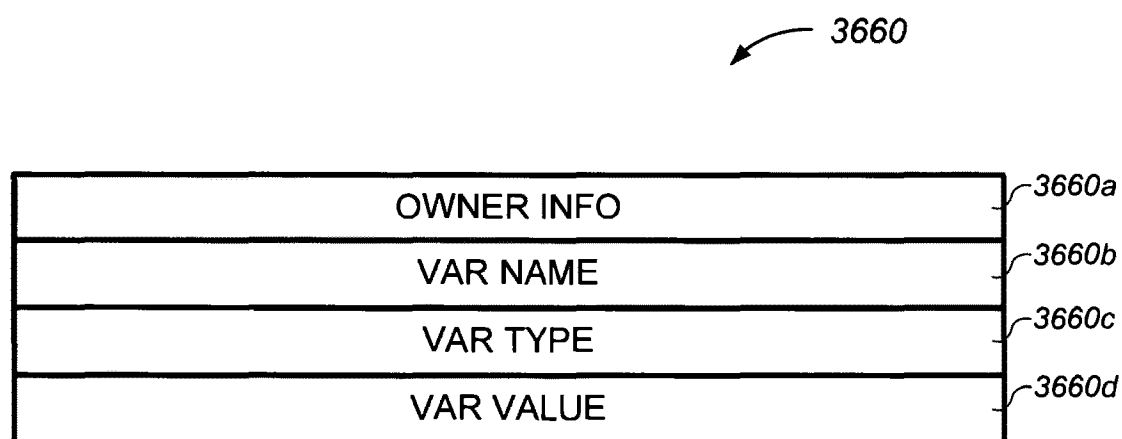


Fig. 36D

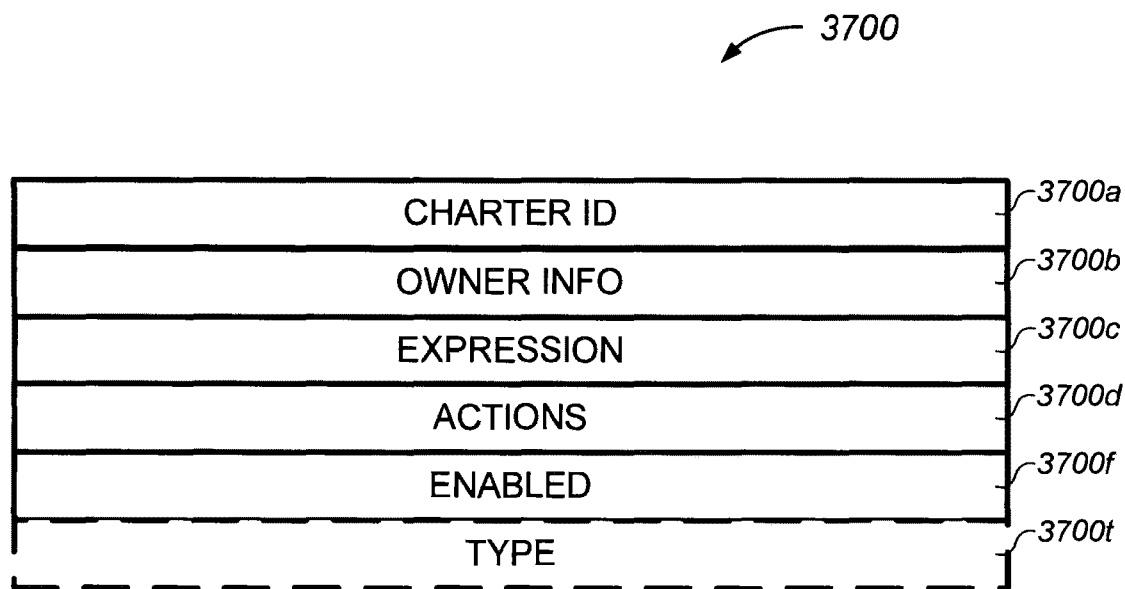
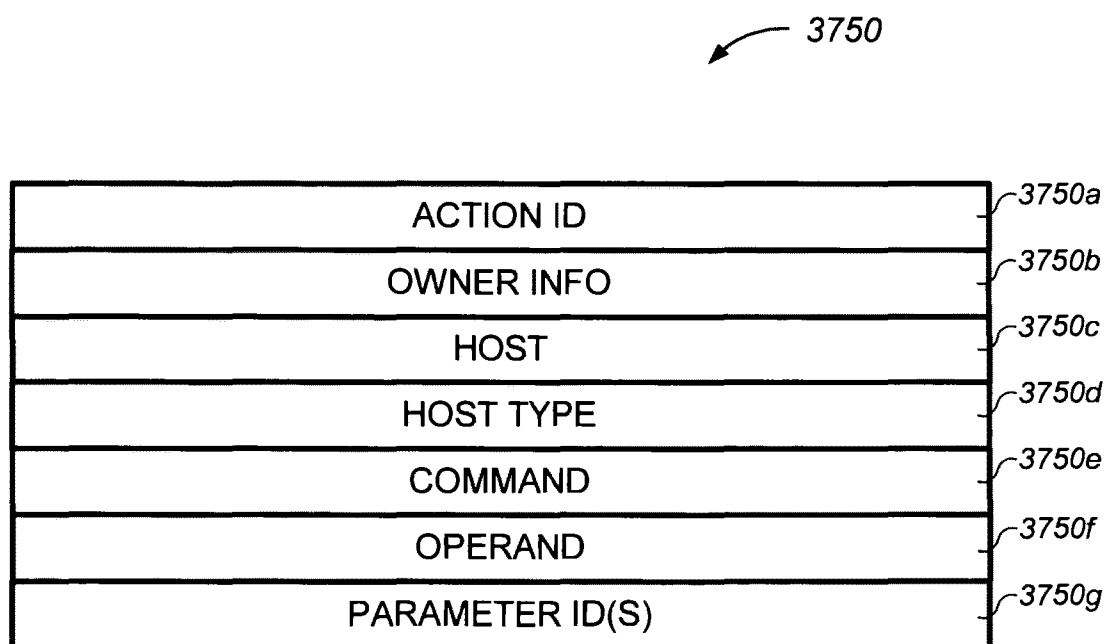


Fig. 37A

**Fig. 37B**

U.S. Patent

Nov. 19, 2019

Sheet 107 of 322

US 10,477,994 B2

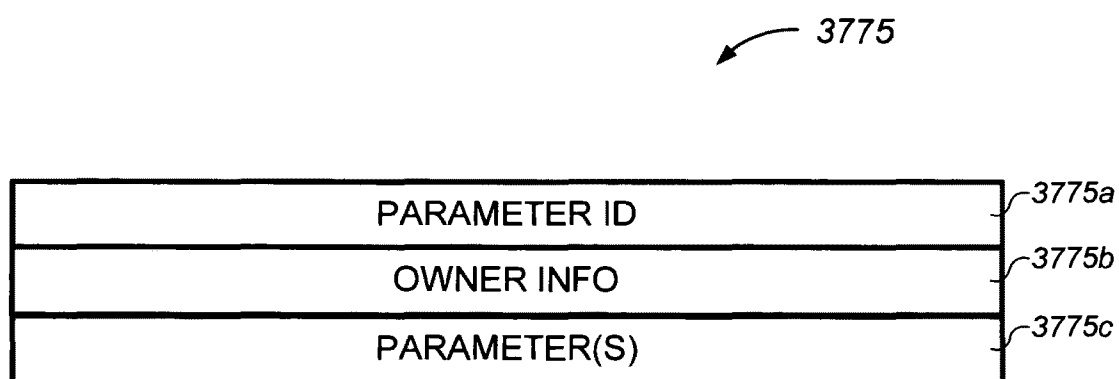
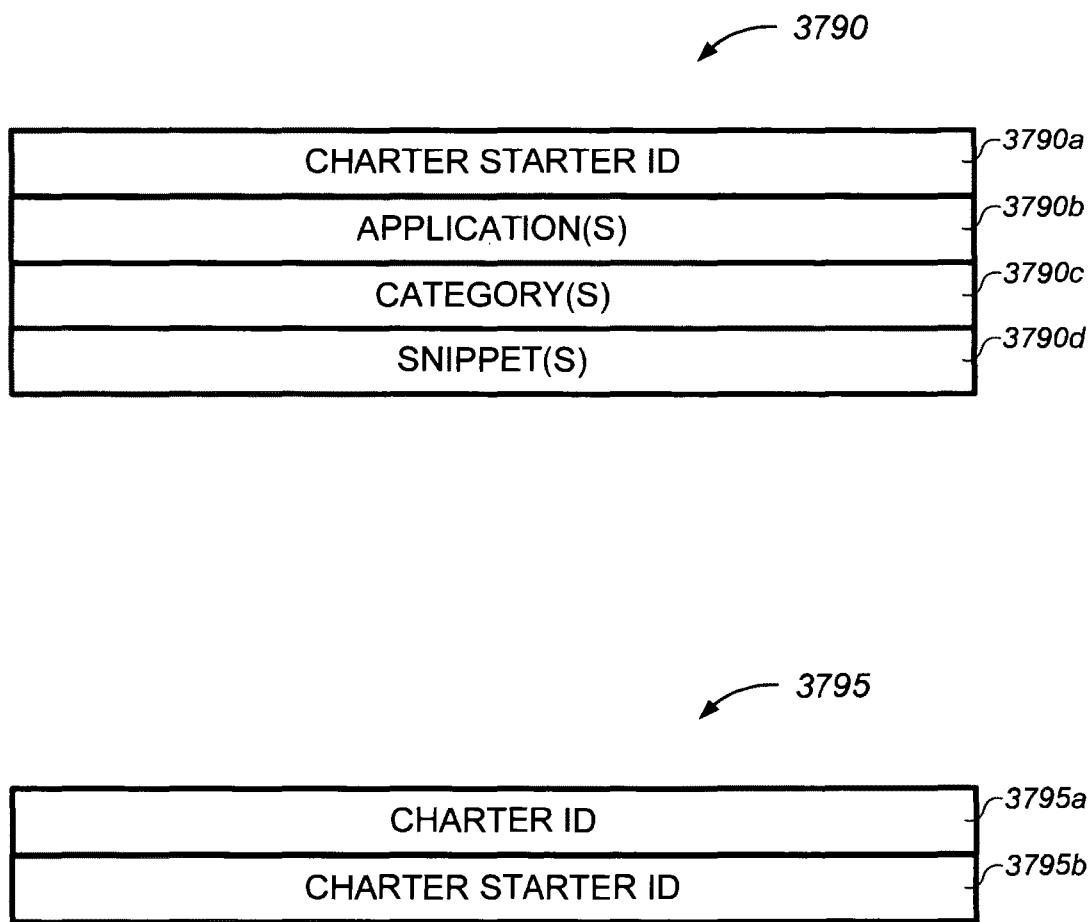


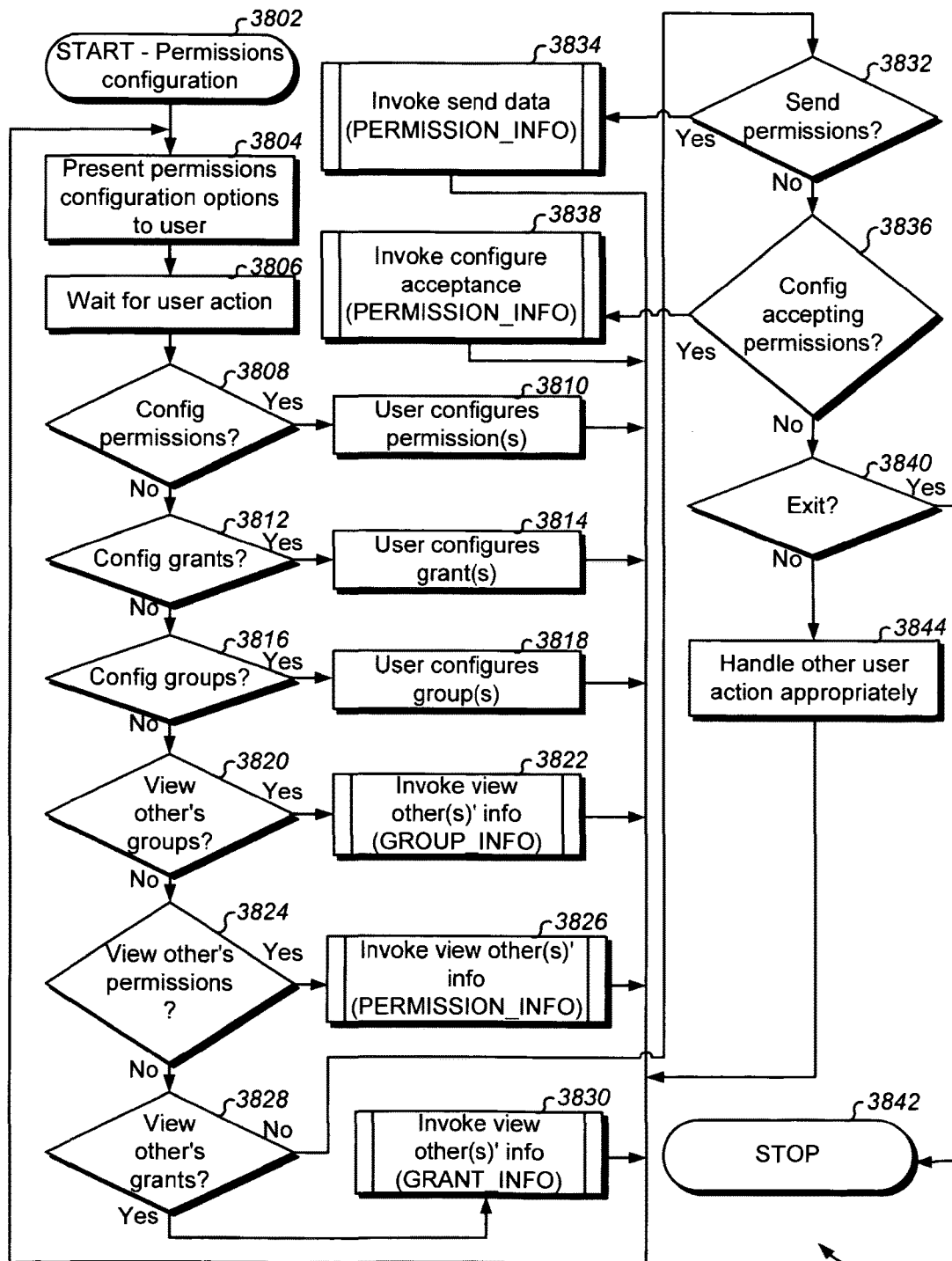
Fig. 37C

U.S. Patent

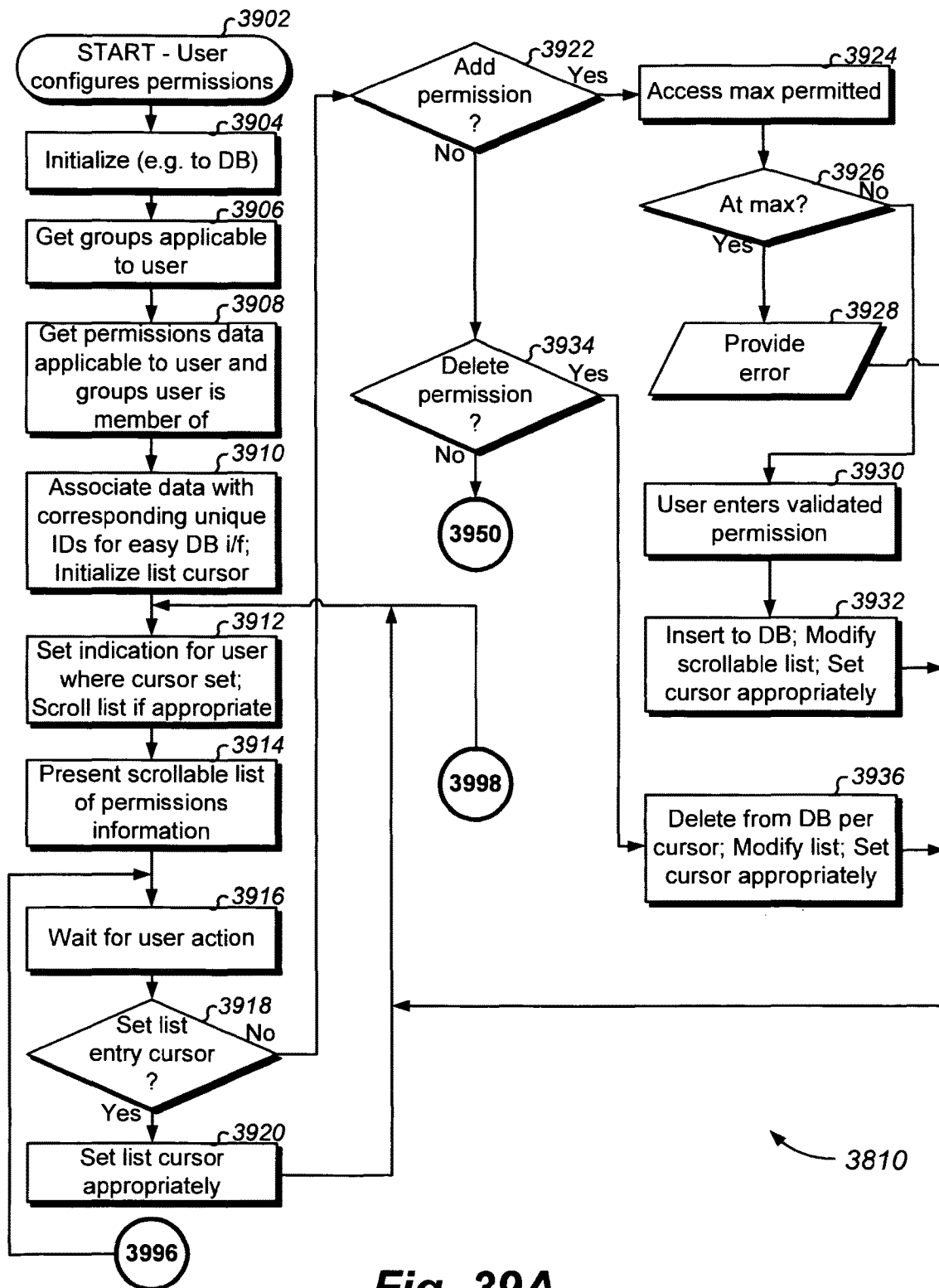
Nov. 19, 2019

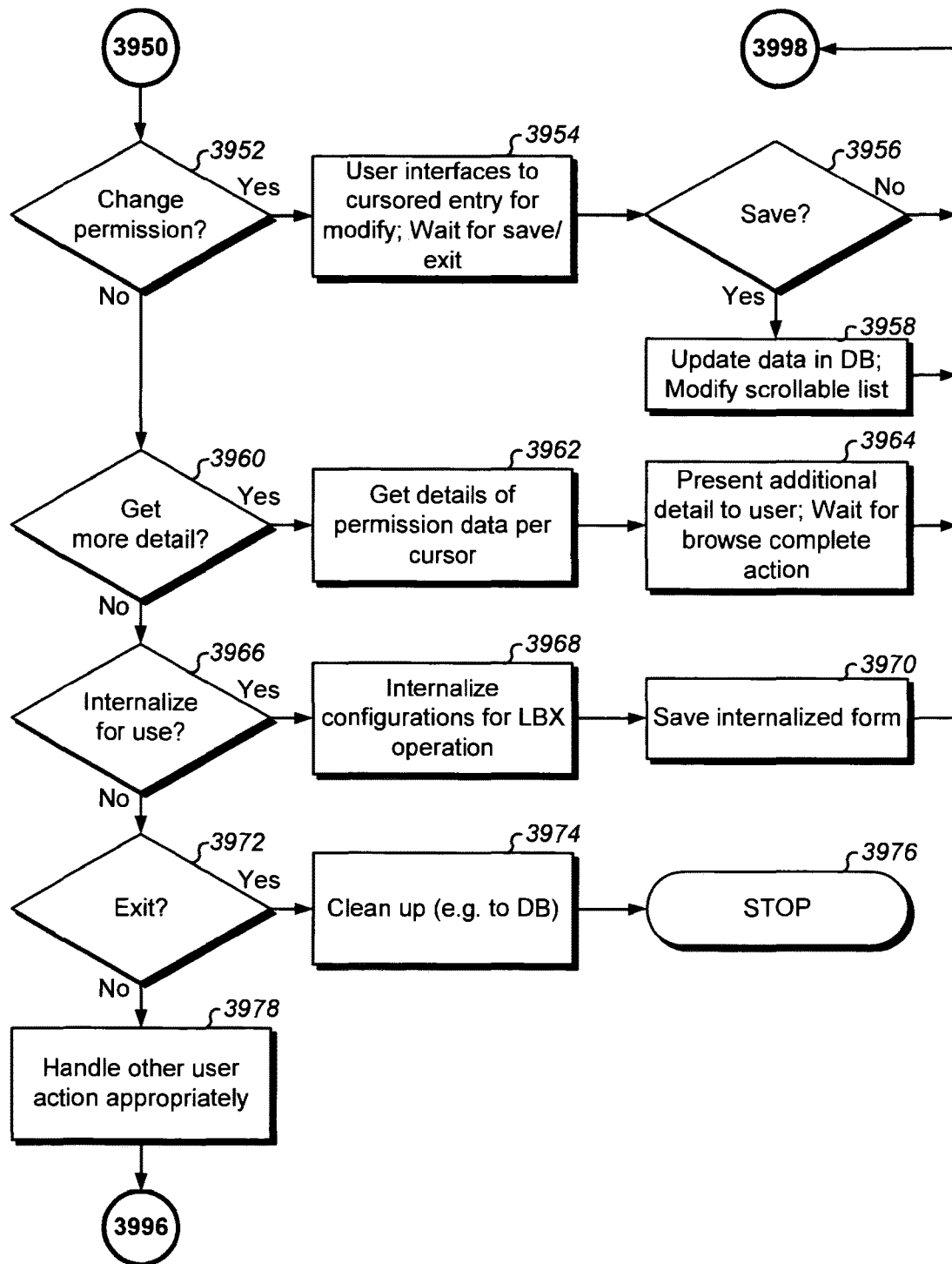
Sheet 108 of 322

US 10,477,994 B2**Fig. 37D**

**Fig. 38**

1478

**Fig. 39A**

**Fig. 39B**

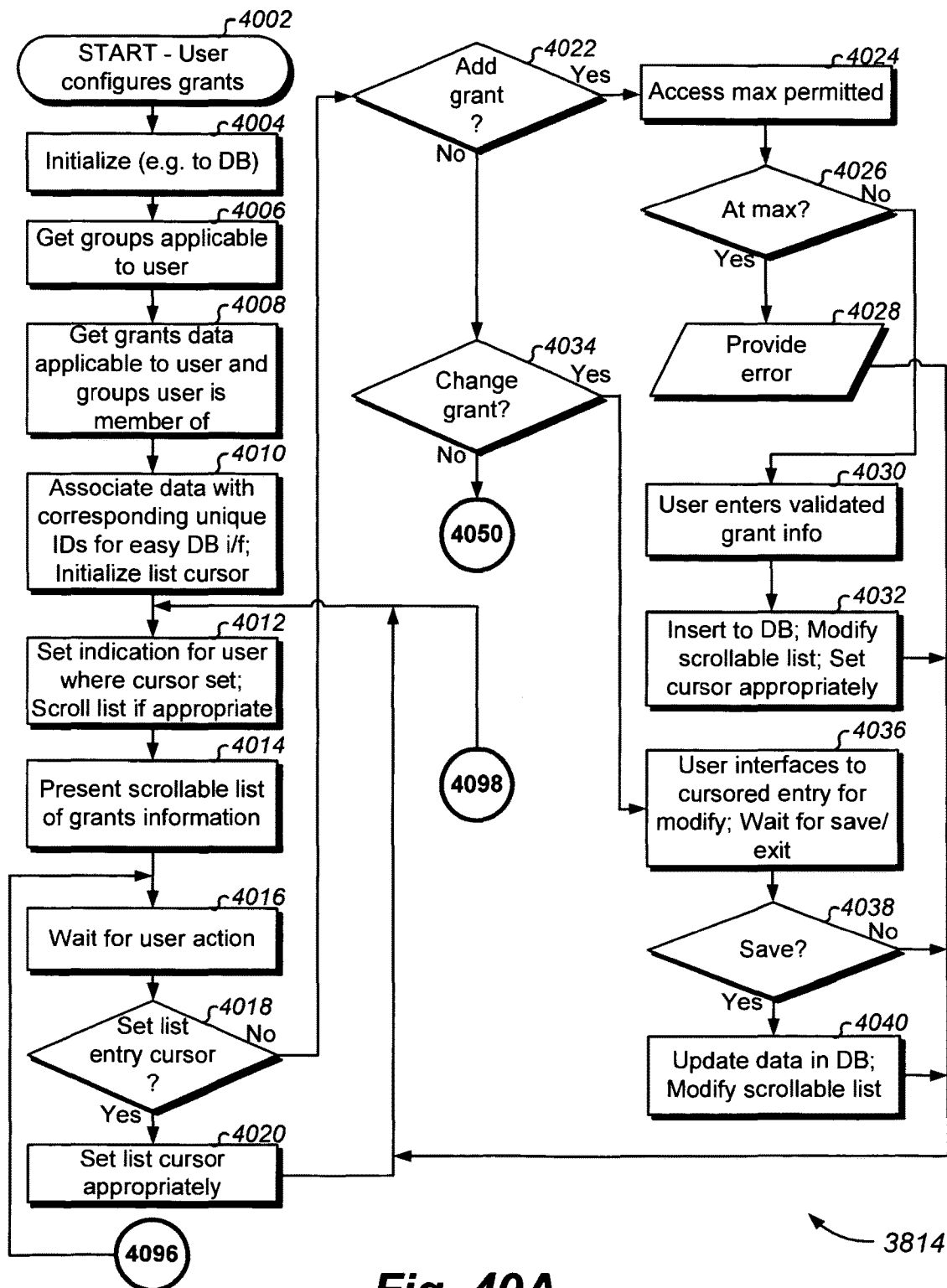
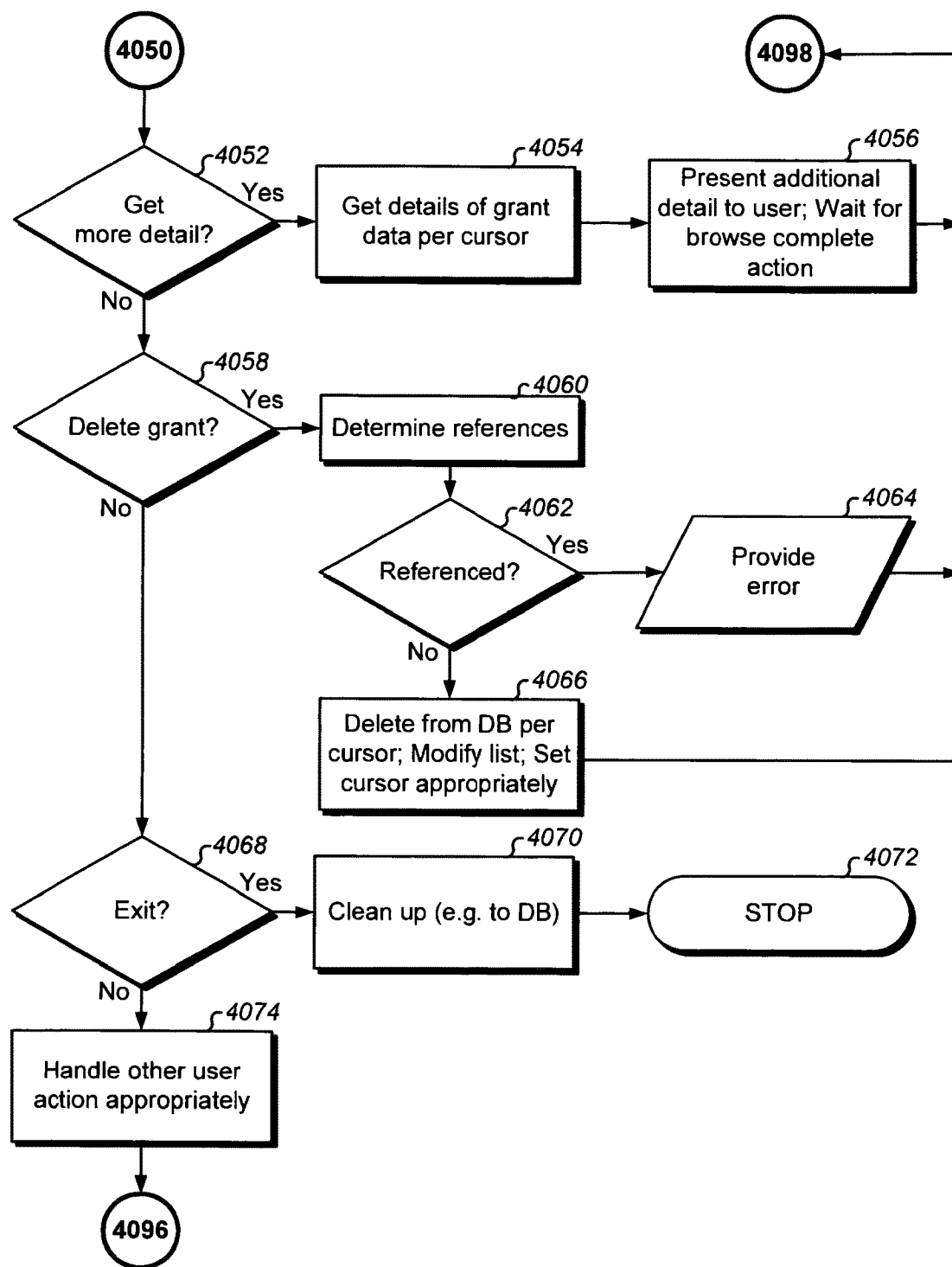


Fig. 40A

**Fig. 40B**

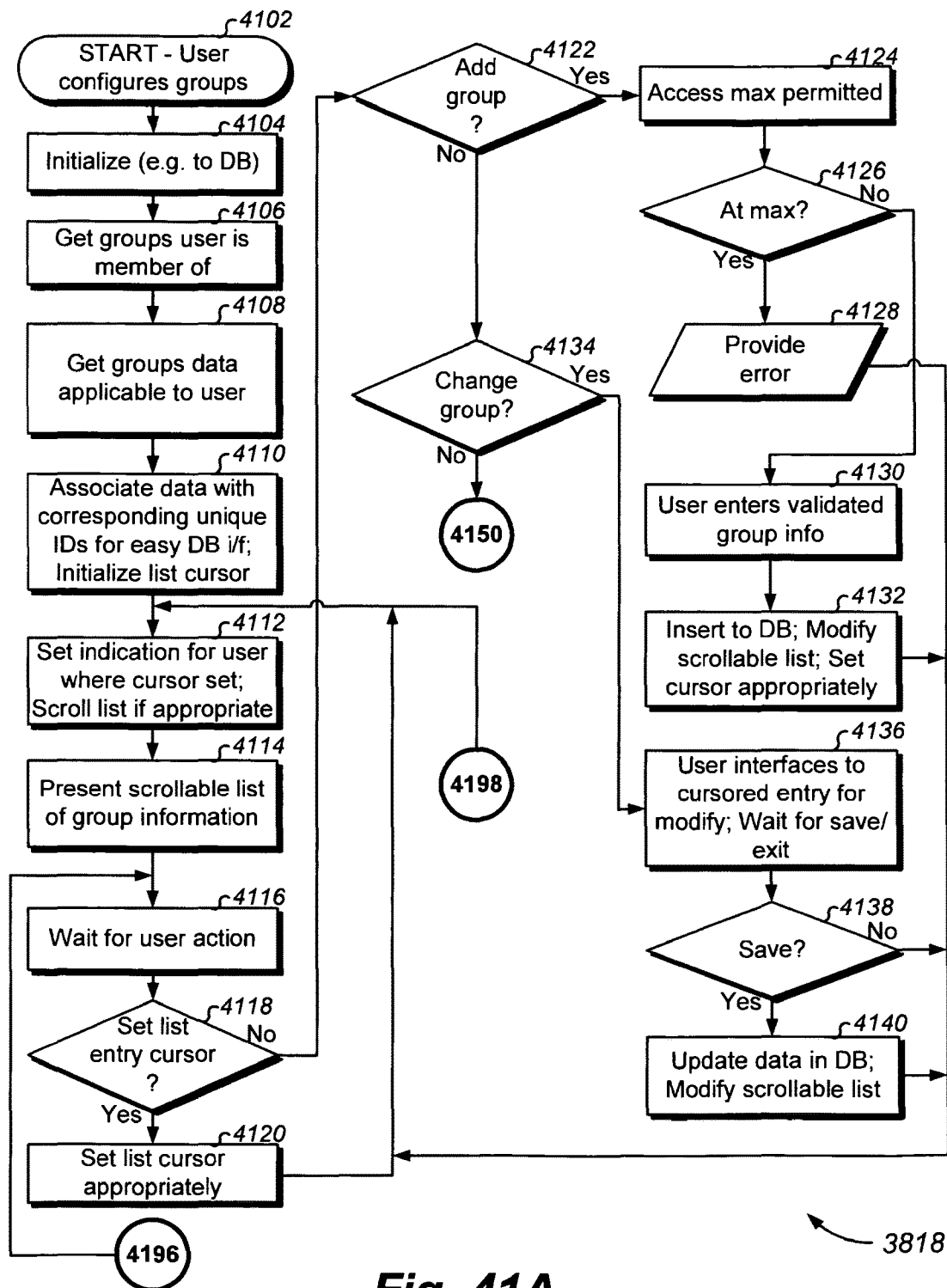
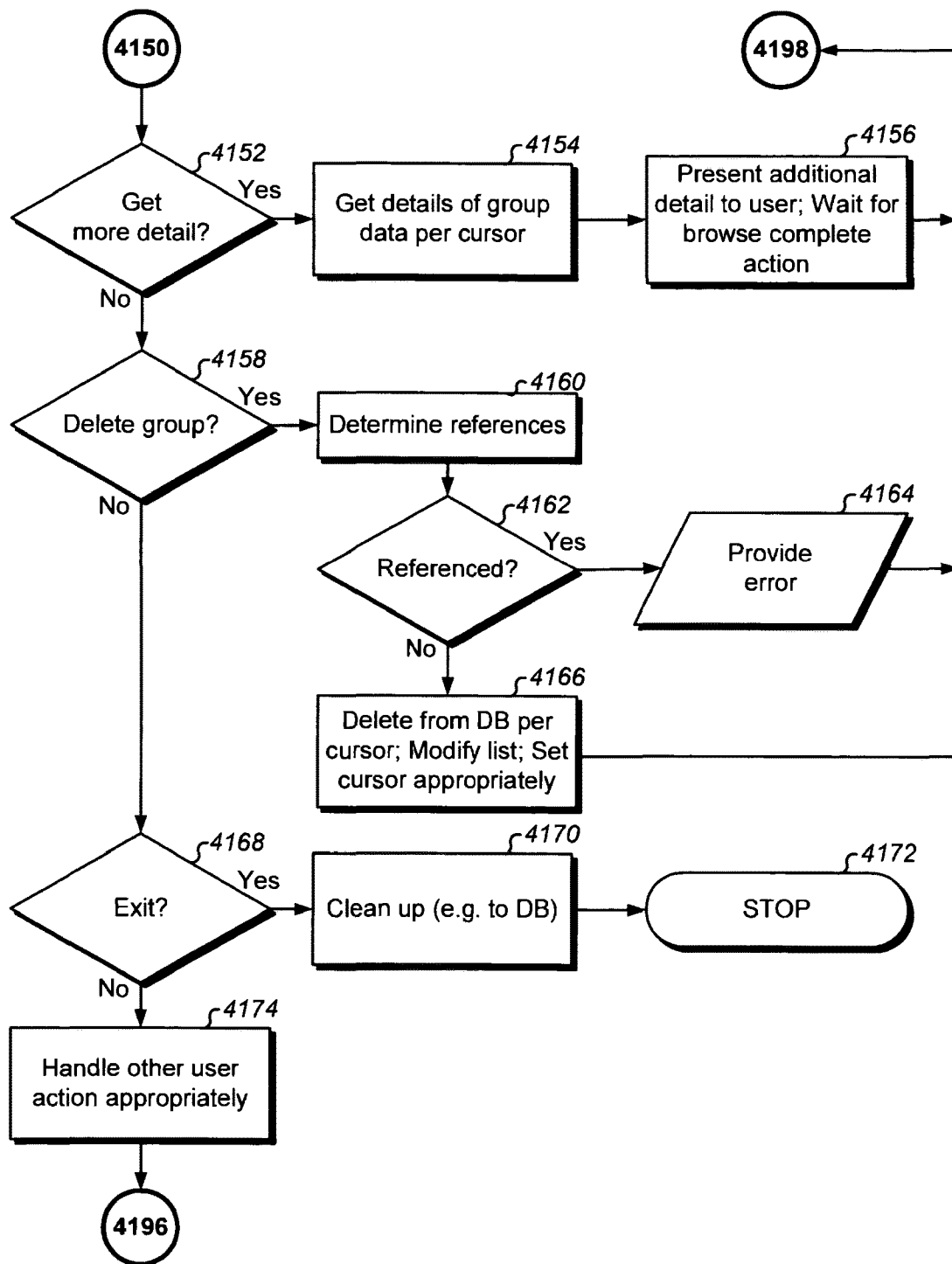
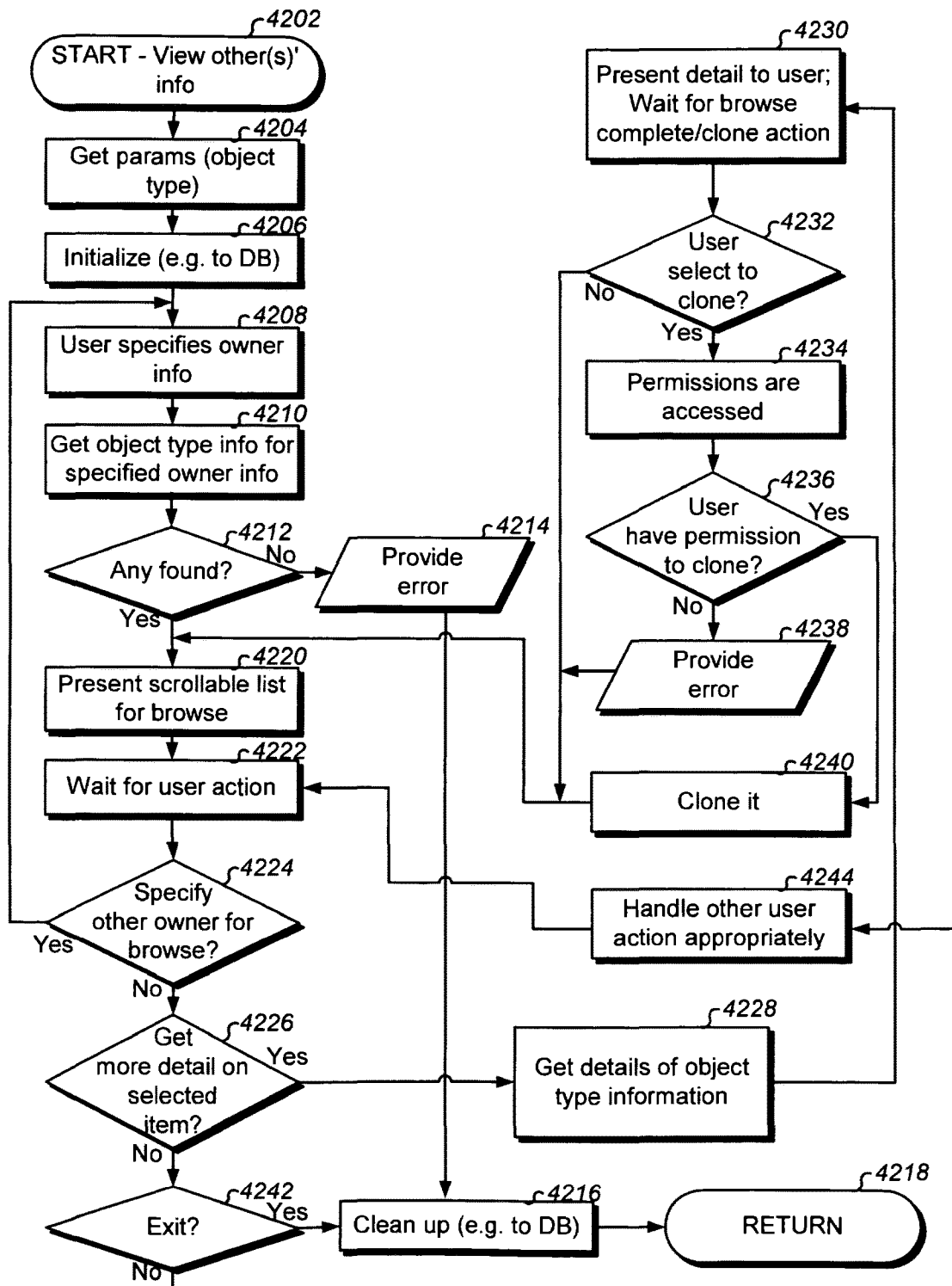
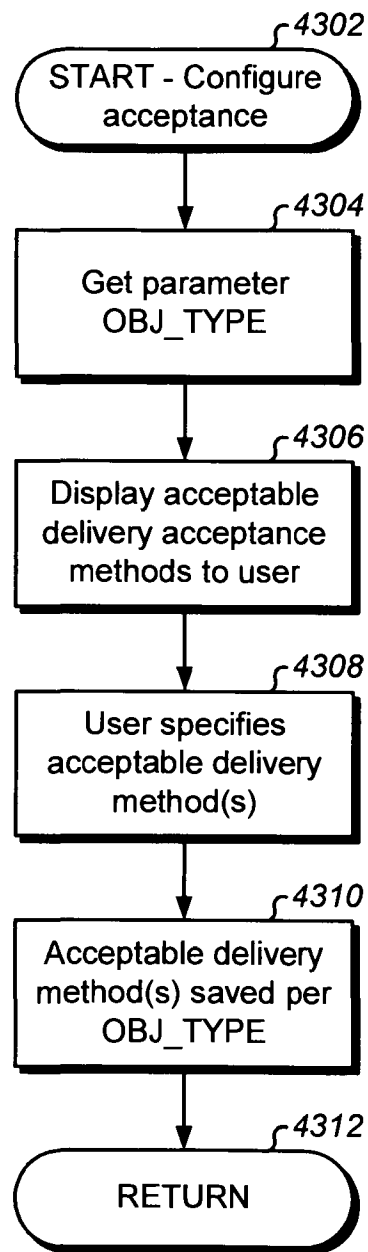
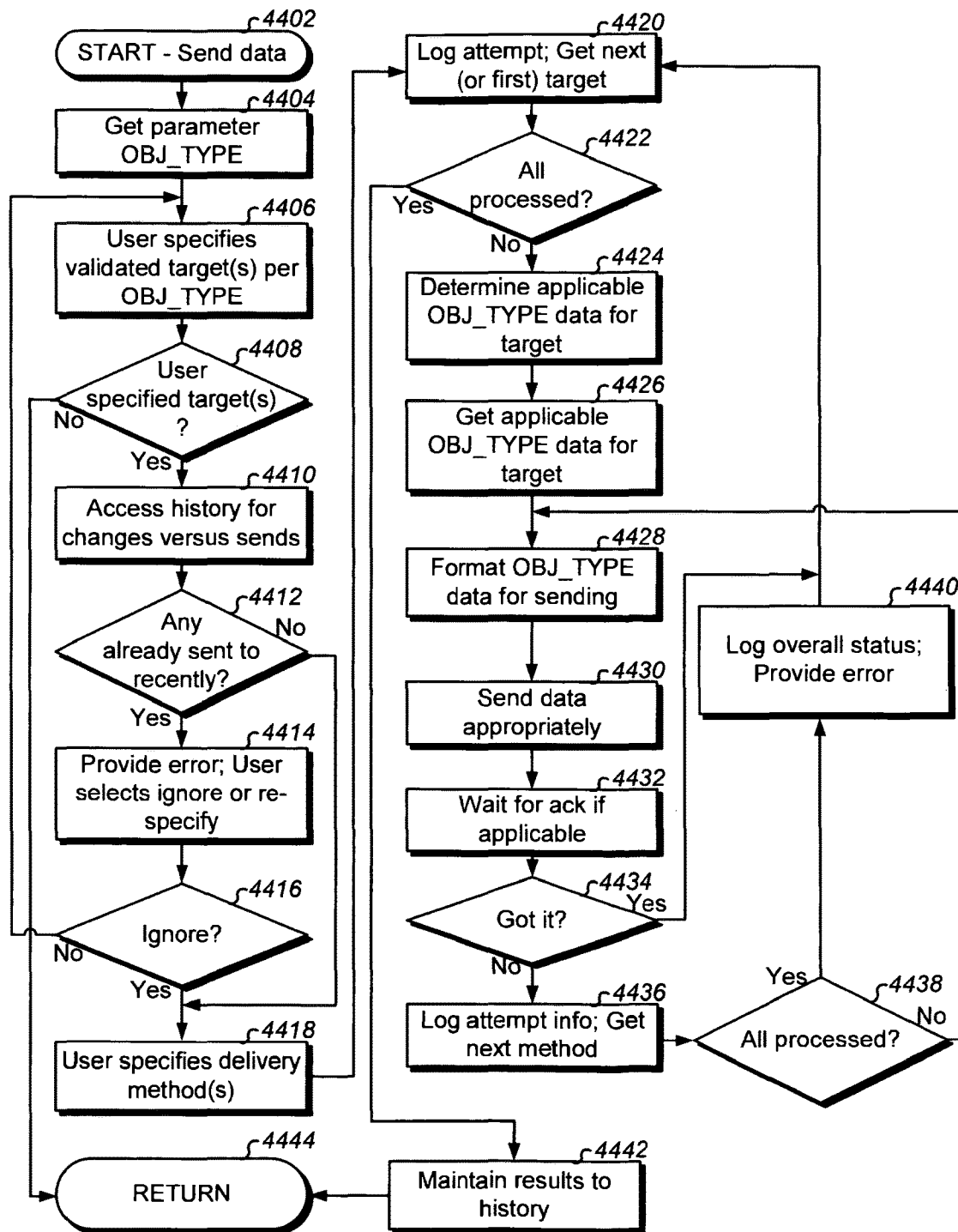


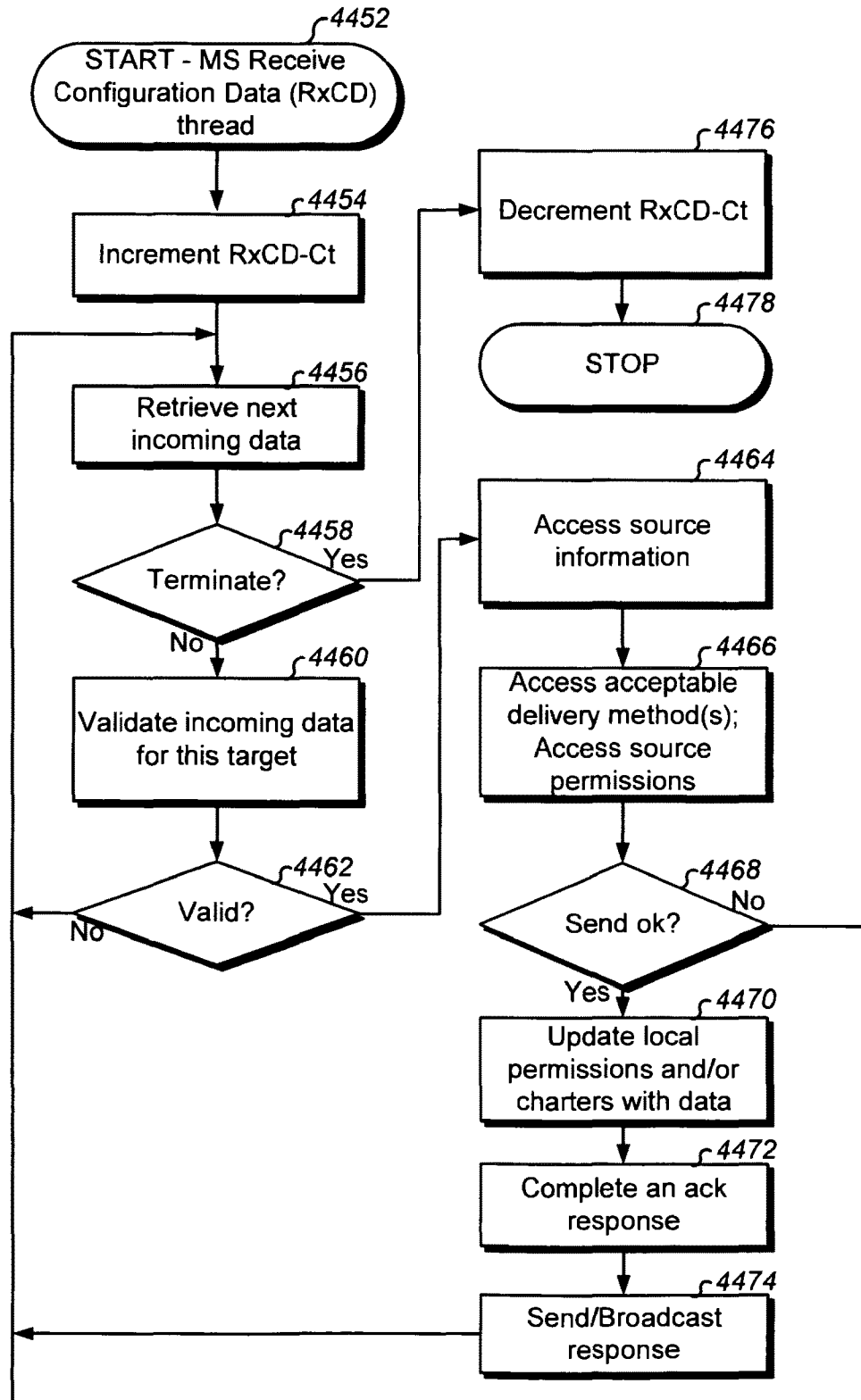
Fig. 41A

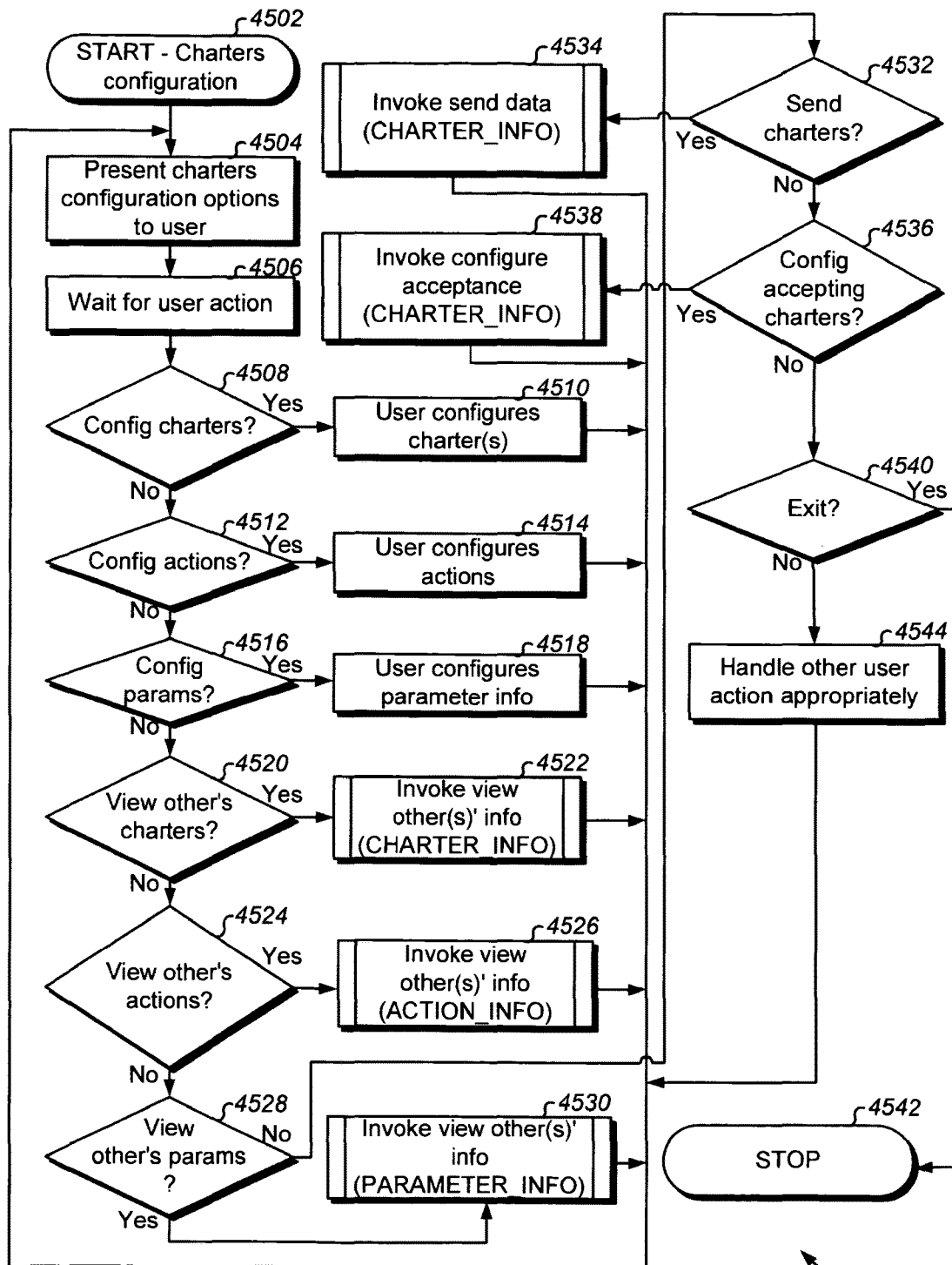
**Fig. 41B**

**Fig. 42**

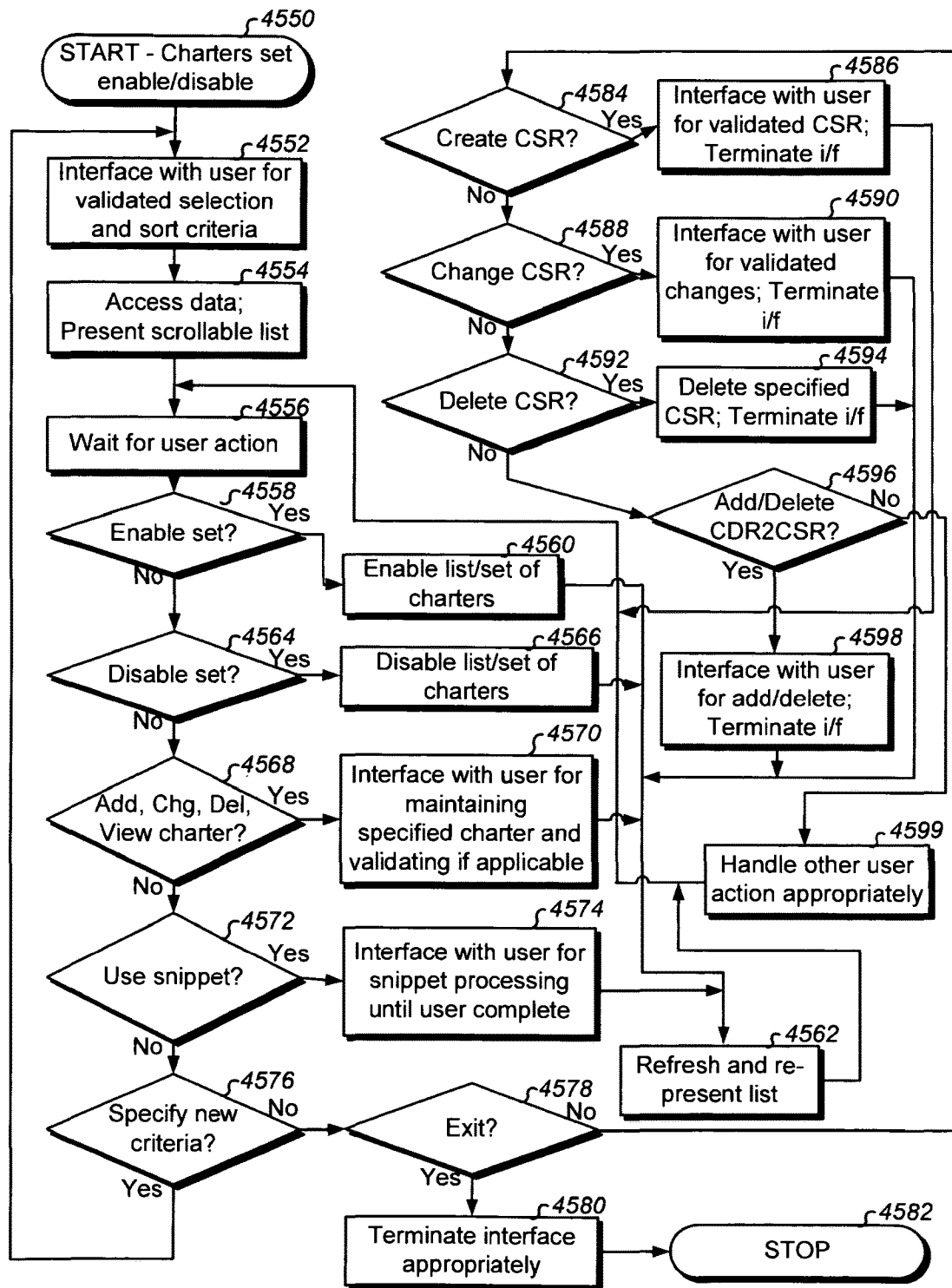
**Fig. 43**

**Fig. 44A**

**Fig. 44B**

**Fig. 45A**

1482

**Fig. 45B**

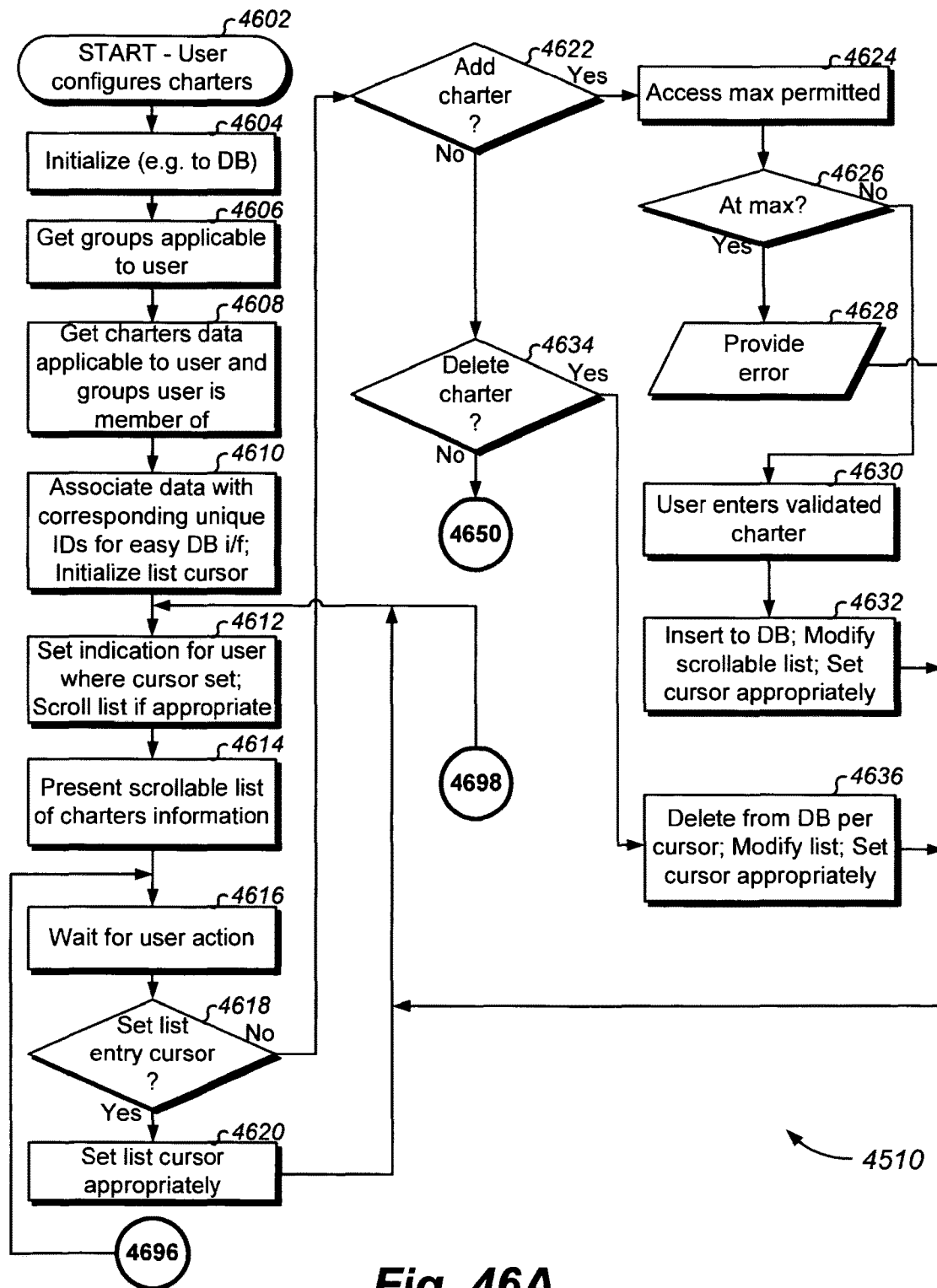
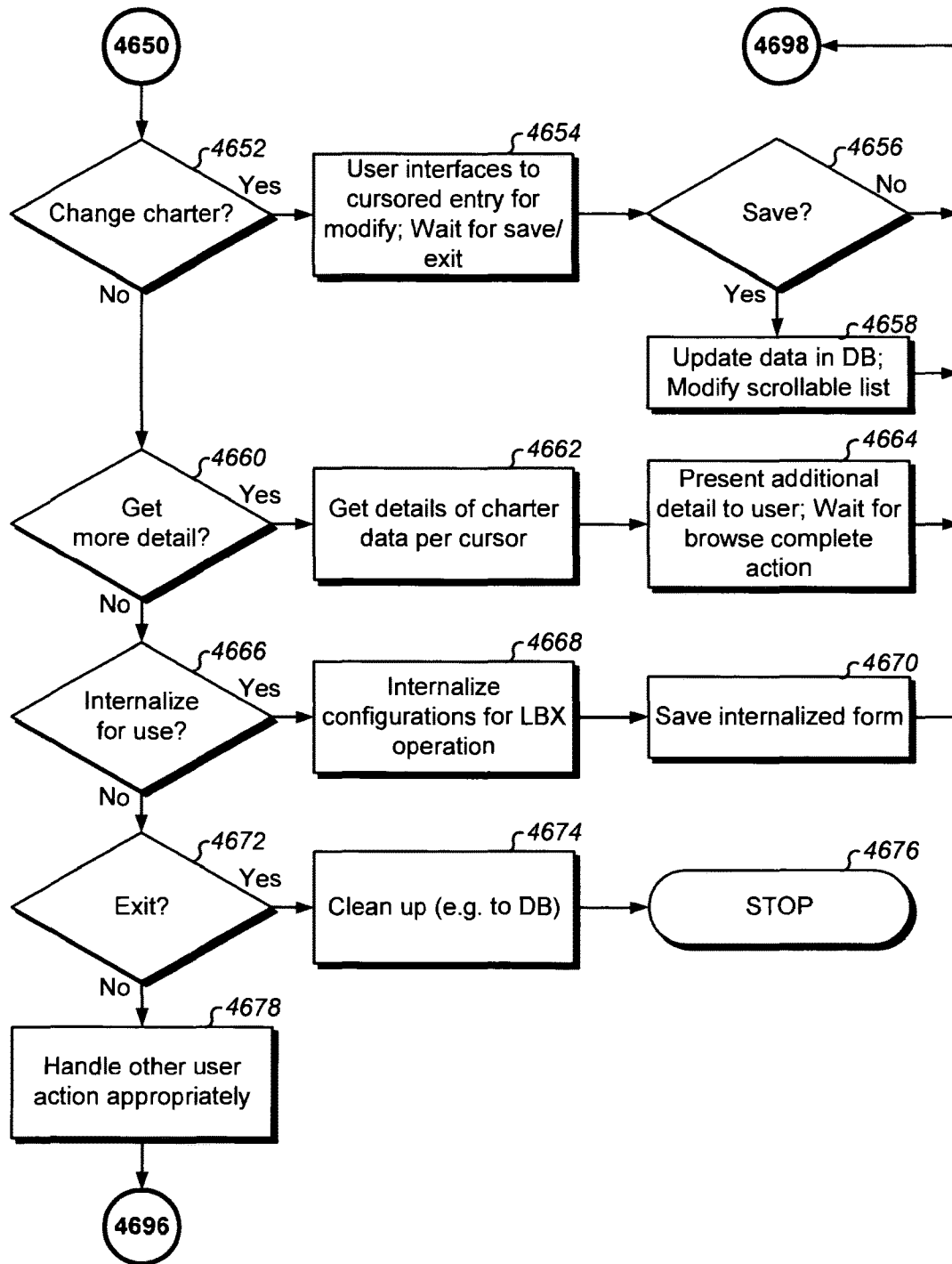


Fig. 46A

**Fig. 46B**

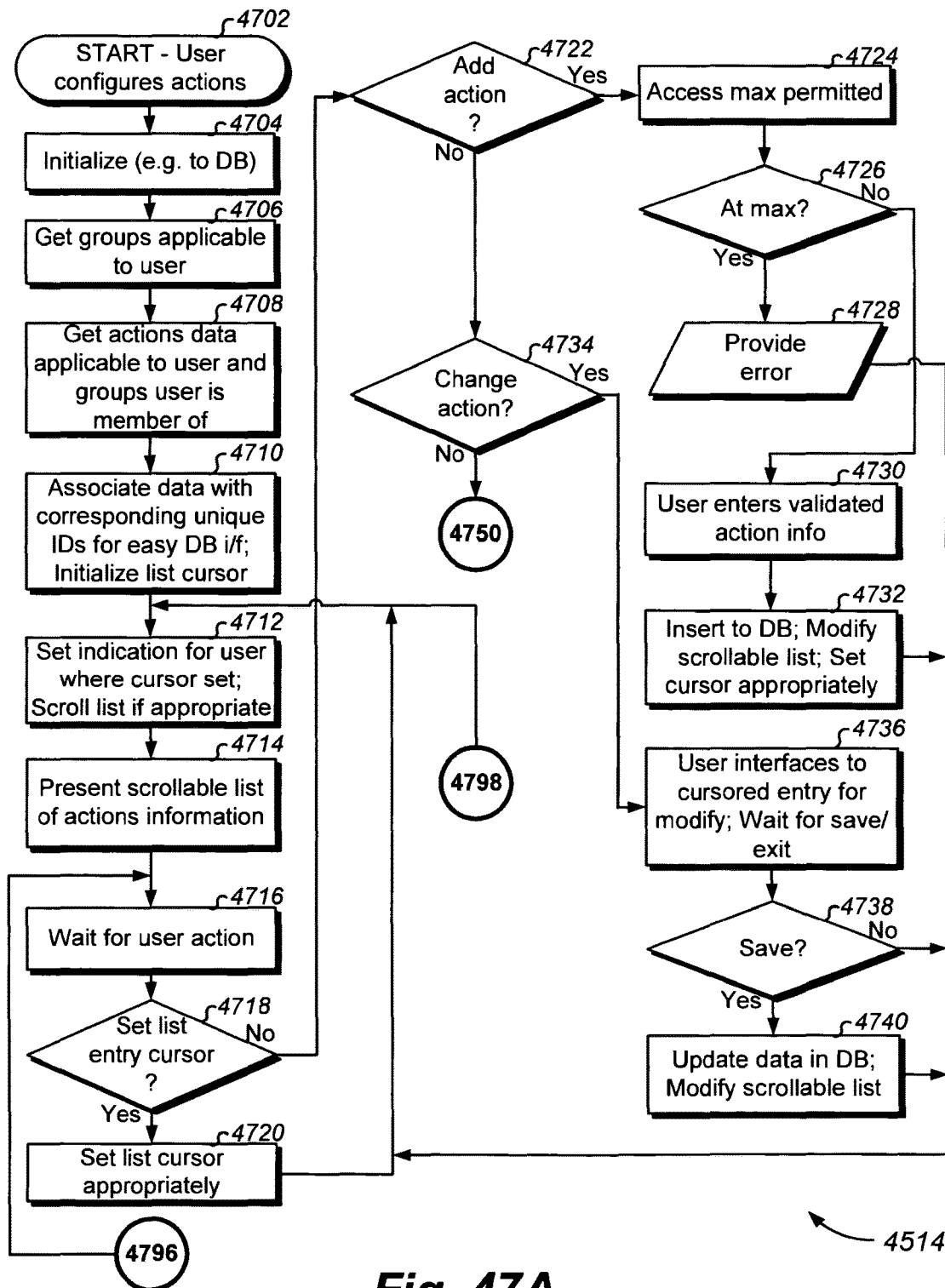
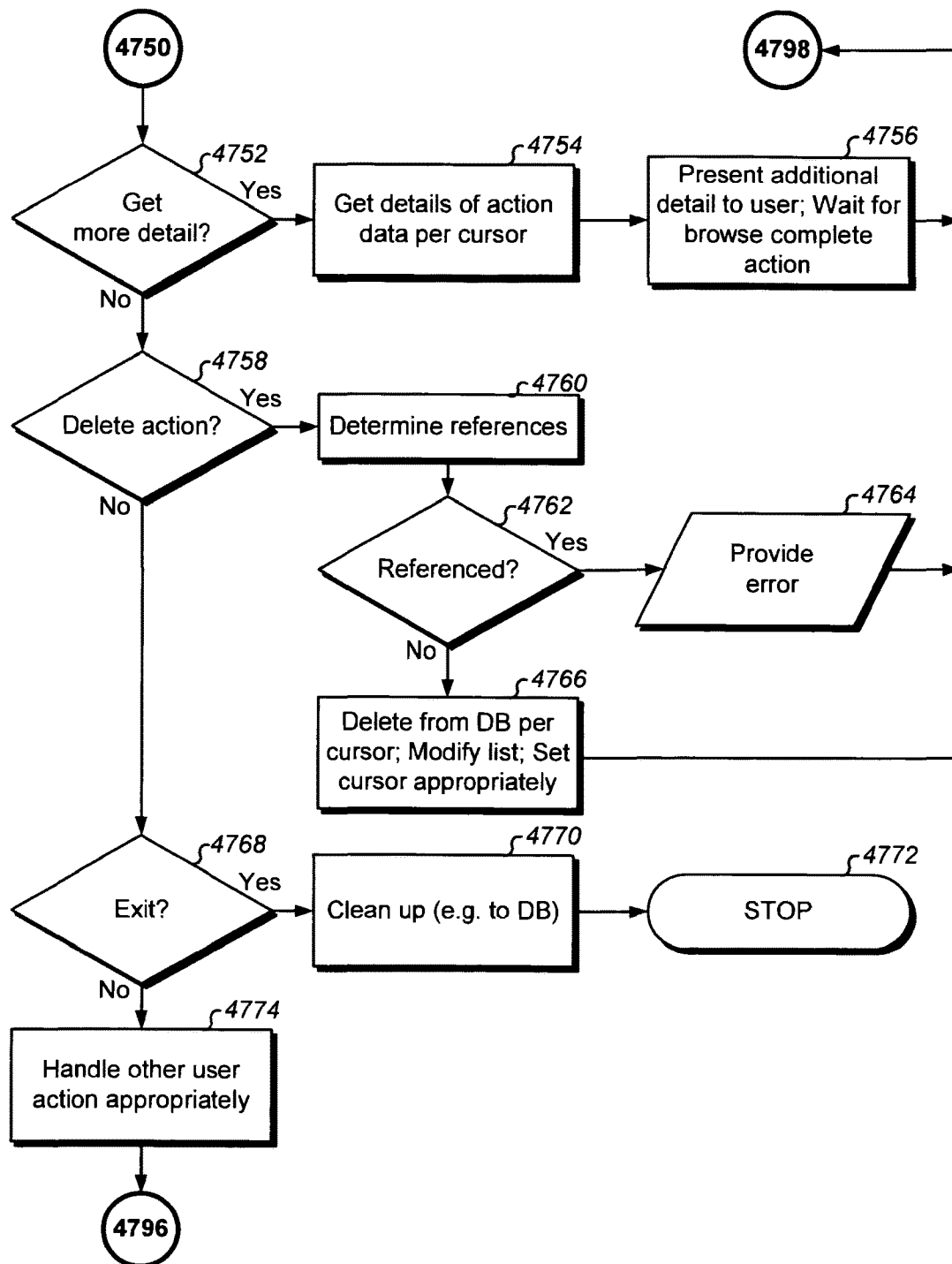


Fig. 47A

**Fig. 47B**

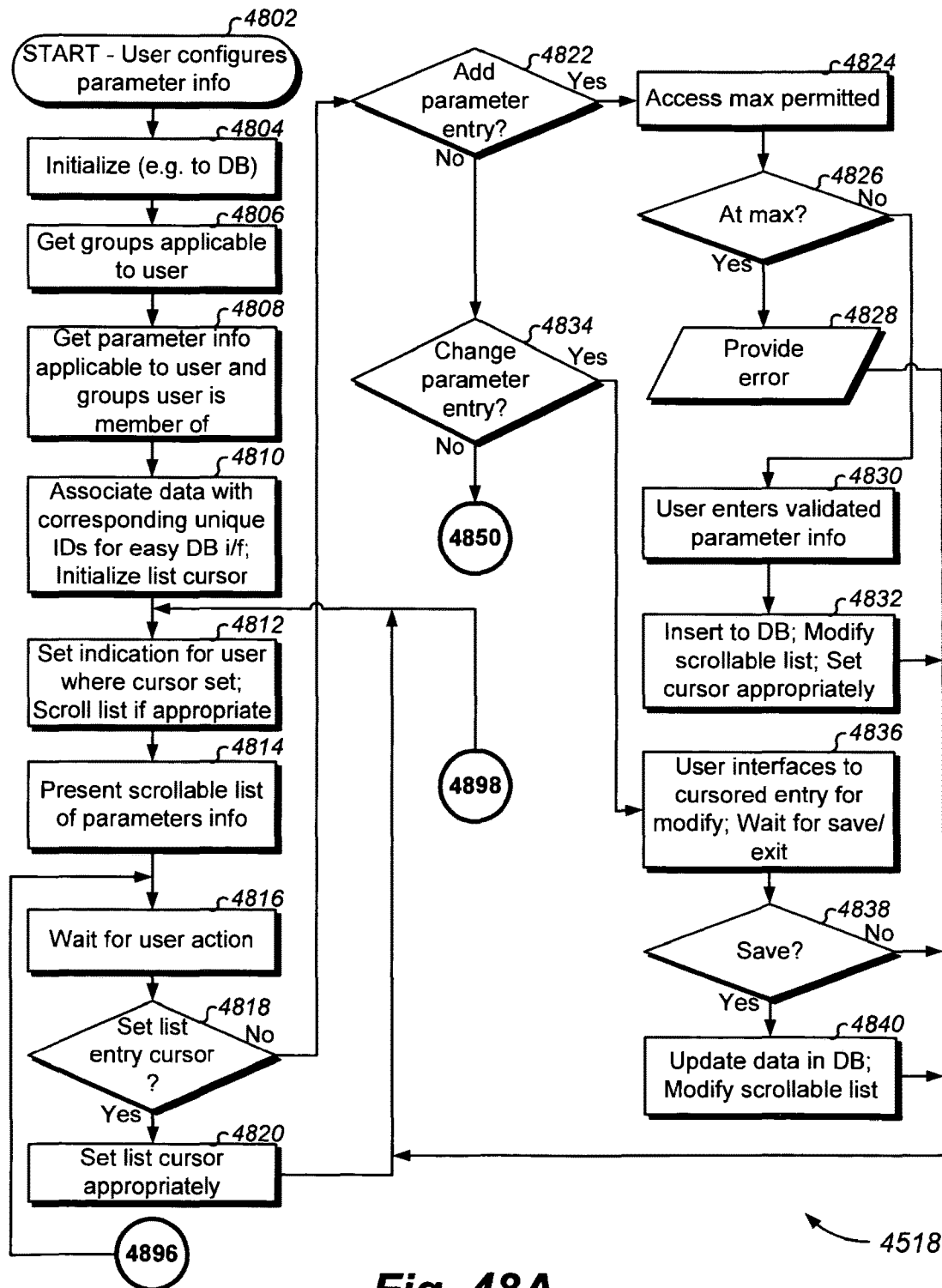
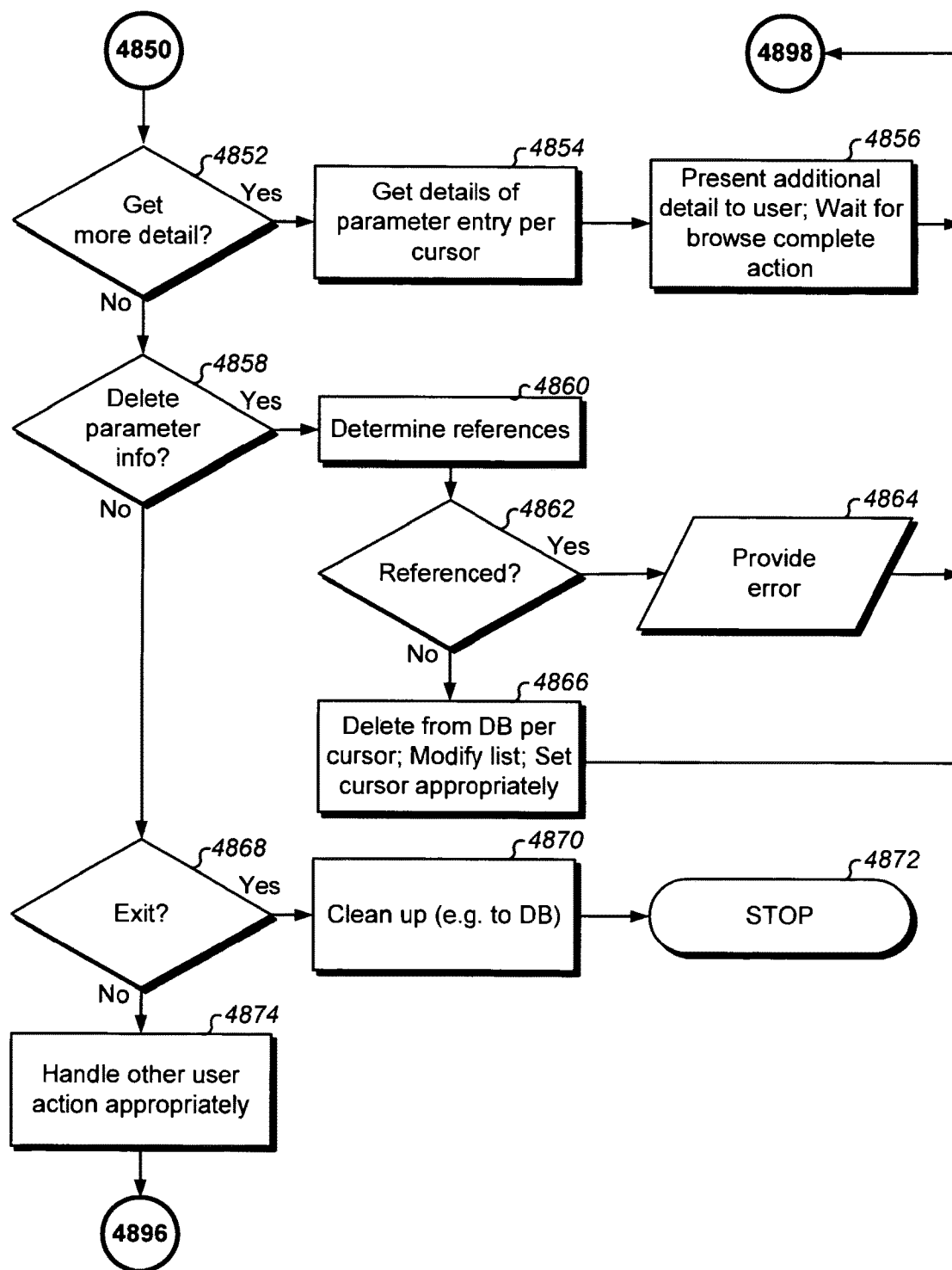
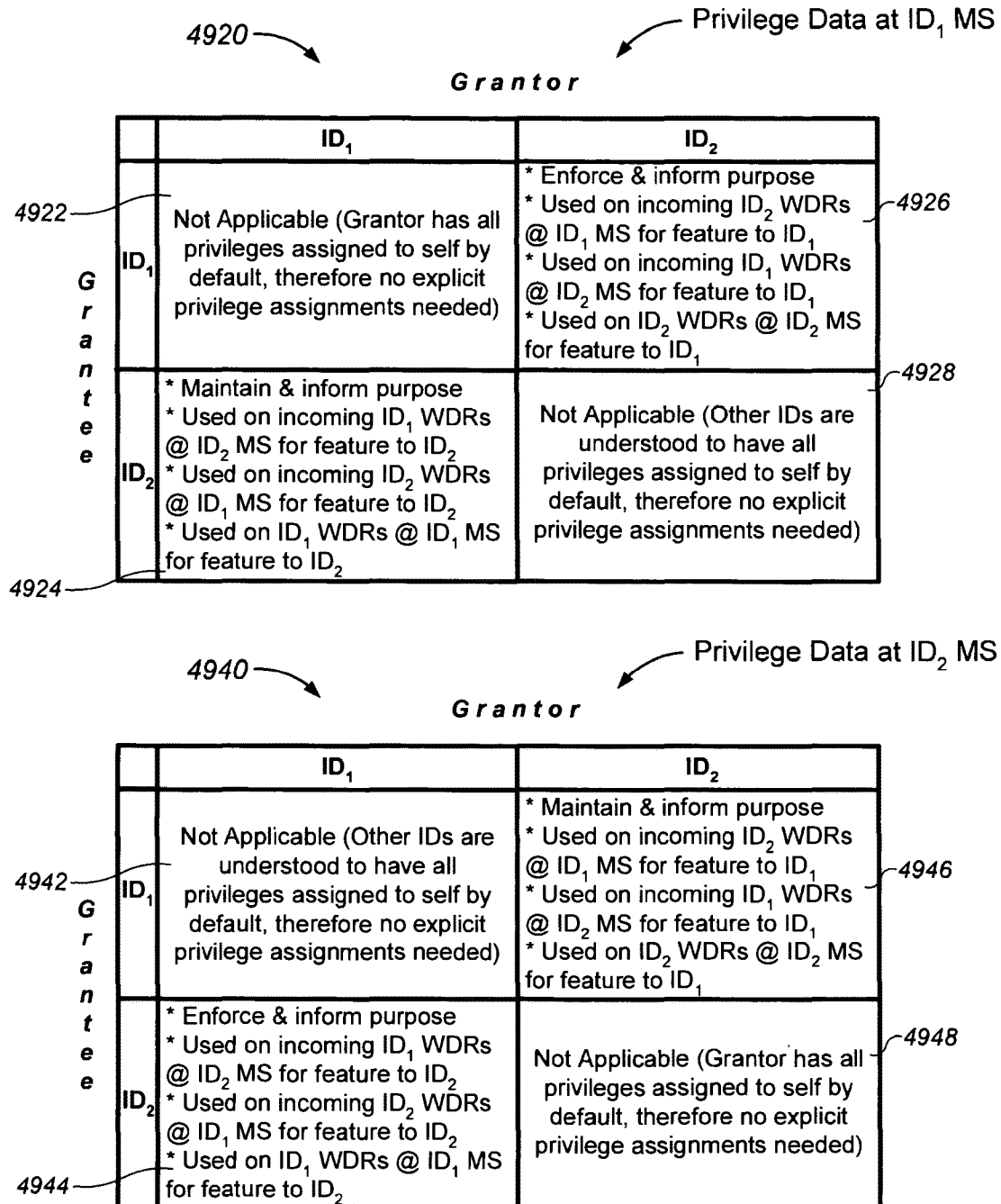


Fig. 48A

**Fig. 48B**

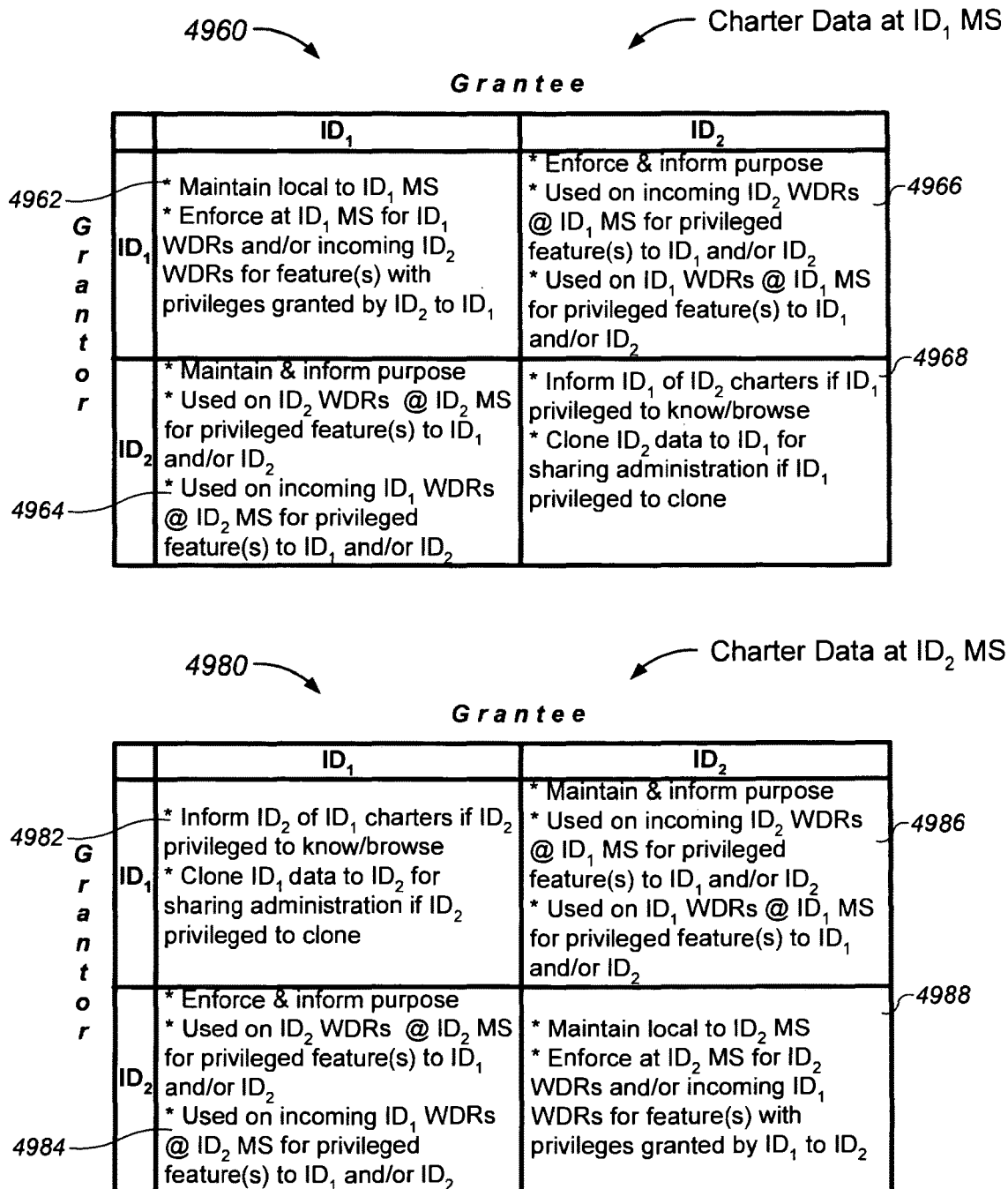
**Fig. 49A**

U.S. Patent

Nov. 19, 2019

Sheet 129 of 322

US 10,477,994 B2

**Fig. 49B**

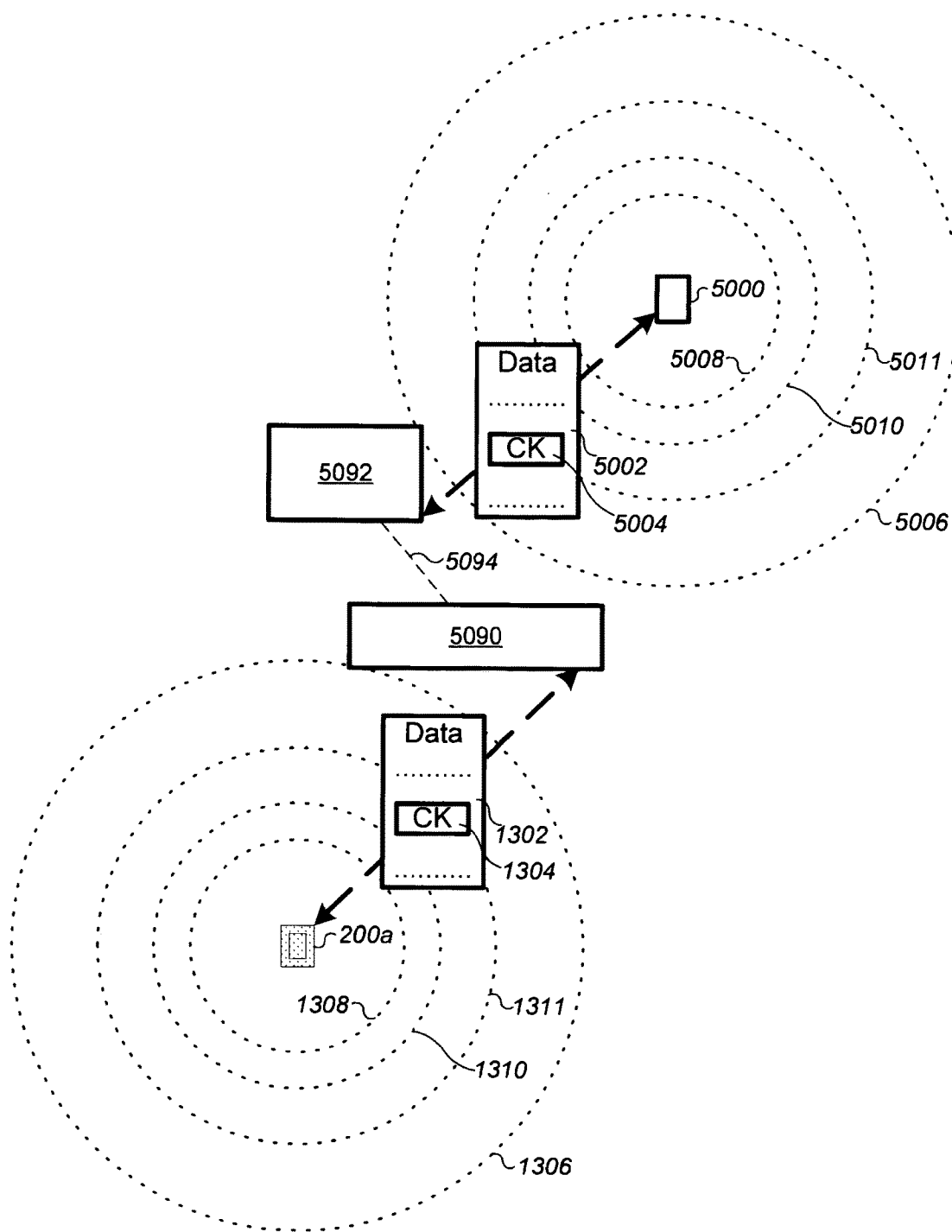


Fig. 50A

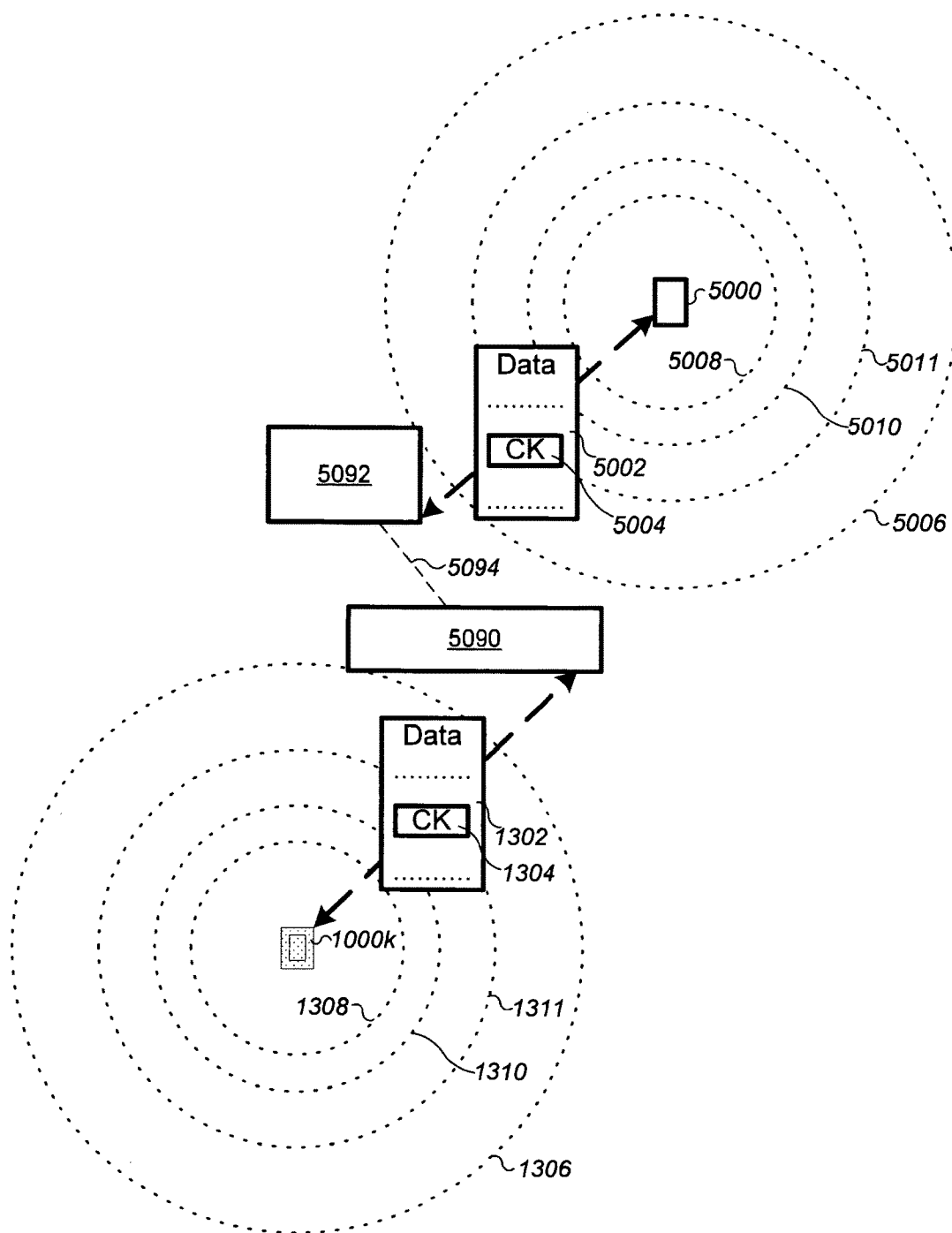


Fig. 50B

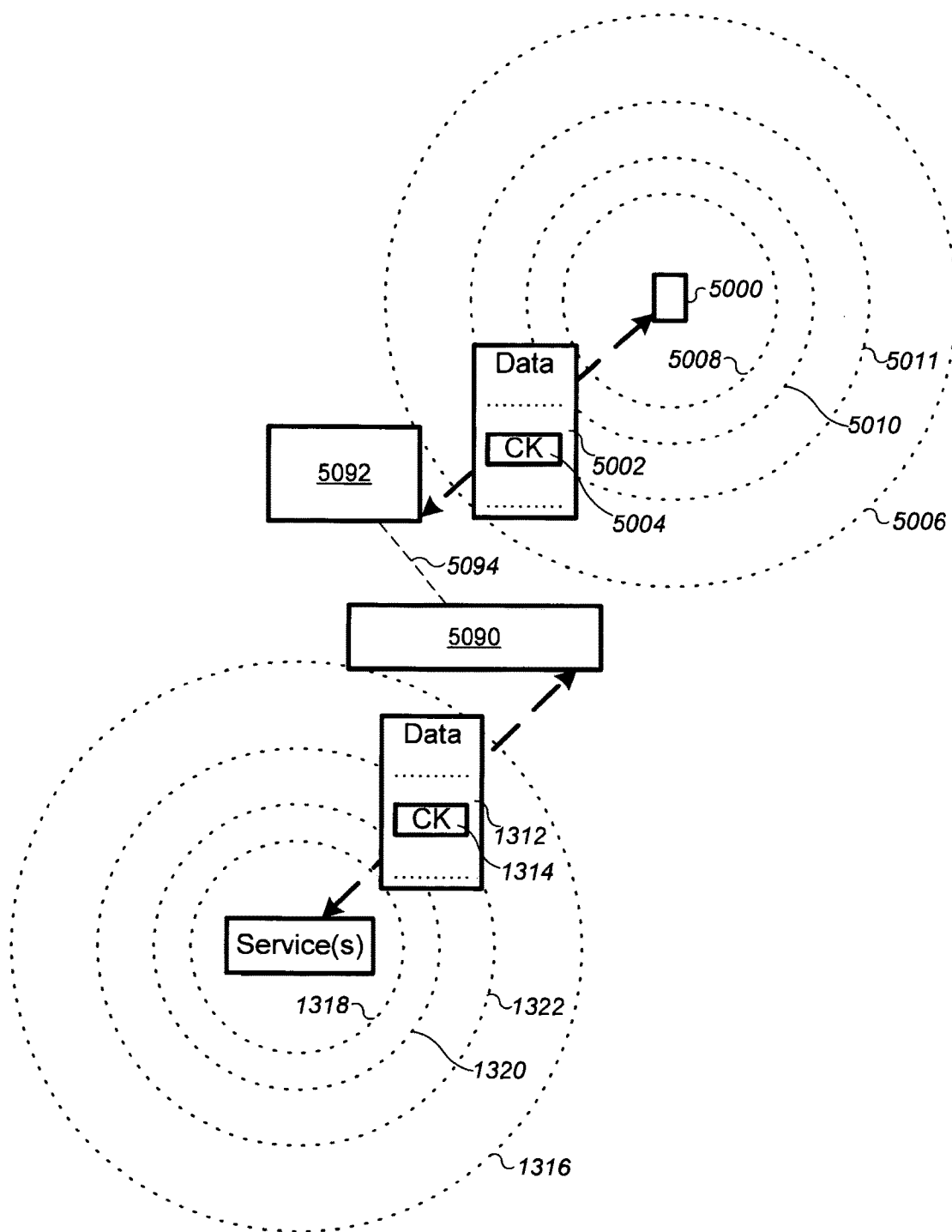


Fig. 50C

```

Permissions {

    Text(str) = "Test Case #106729 (context)";
    Generic(assignPrivs) = "G=Family,Work,\vuloc [T=>20080402000130.24,<20080428;
        D=*str; H;]";
    Groups {
        LBXPHONE_USERS = Austin, Davood, Jane, Kris, Mark, Ravi, Sam, Tim;
        "SW Components" = "SM 1.0", "PIP 1.0", "PIPGUI 1.0", "SMGUI 1.0",
            "COMM 1.0", "KERNEL 1.1";
    }

    Grants /* Can define Grant structure(s) prior to assignment */ {
        Family= \lbxall[R=0xFFFFFFFF;] [D=*str(context="Family")];
        Work = [T=YYYYMMDD08:YYYYMMDD17;D=*str(context="Work");H;] {
            "Department 232"=\geoar,\geode,\nearar,\nearde;
            "Department 458" = [D="Davood lyadi's mgt scope";] {
                "Server Development Team" = ,
                "lbxPhone Development Team" = {
                    "Comm Layer Guys" = \mssys;\msbios;
                    "GUI girls" = \msguiload;
                    "Mark and Tim" = \msapps;
                };
            };
            "Accounting Department" [H;] = \track;
        };
        Parents = { Mom=\lbxall; Dad=\lbxall; };
        Michael-Friends=\geoarr,\geode;
        Jason-Friends=\nearar,\nearde;
    }

    // Permissions are granted here:
    Bill: LBXPHONE_USERS [G=\caller;\callee;\trkall;];
    LBXPHONE_USERS: Bill [G=\callee;\caller;];
    Bill:Sophia;
    Bill:Brian [*assignPrivs];
    Bill:George [G=\geoall,\nearall;];
    Michael Bill [G=Parents,Michael-Friends;];
    Jason: Bill [G=Parents,Jason-Friends;];
}

```

Fig. 51A

Charters {

```

Condition(cond1) = "(__location @@ \loc_my) [D="Test Case #104223 (v)";];
"ms group" = { "Jane", "George", "Sally" };

( (__msid = "Michael") & *cond1(v="Michael")) |
  ((__msid = "Jason") & *cond1(v="Jason")) ):
  Invoke App myscript.cmd ("S"), Notify Autodial 214-405-6733;

(_I_msid = "Brian") & (_I_location @ \loc_my) [D="multi-cond text";H;]:
  Invoke App (myscript.cmd ("B")) [T=20080302;],
  Notify Autodial (214-405-5422);

(M_sender = ~emailAddrVar [T=<YYYYMMDD18]):
  Notify Indicator (M_sender, \thisMS) [D="Test Case #104223"; H;];

(B_srchSubj ^ M_subject) & !(_fcnTest(B_srchSubj)) :
  "ms group"[G].Store DBOject(JOESDB.LBXTABS.TEST,
    "INSERT INTO TABLESAV (" && \thisMS && ", " && \timestamp &&
    ", 9);", \thisMS);

(_I_msid = "Sophia" & \loc_my (30M)$(25M) _I_location ) :
  "ms group".Invoke App (alert.cmd);

(%c:\myprofs\interests.chk > 90):
  Send Email ("Howdy " && _I_msid && " !!\n\nOur profiles matched > 90%.\n\n"
    && "Call me at " && \appfld.phone.id && ". We are " &&
    (_I_location - \loc_my)F && " feet apart\n", \appfld.source.id, "Call Me!",
    , _I_appfld.email.source);

```

}

Fig. 51B

```

typedef struct privilege {
    unsigned long    priv;
    unsigned char    relevance[MAX_RELEVANCEMASK];
    TIMESPEC        *tspec;        // merged with permission level (if permission
                                    // level was present)
    struct privilege *nextPriv;
} PRIVILEGE;

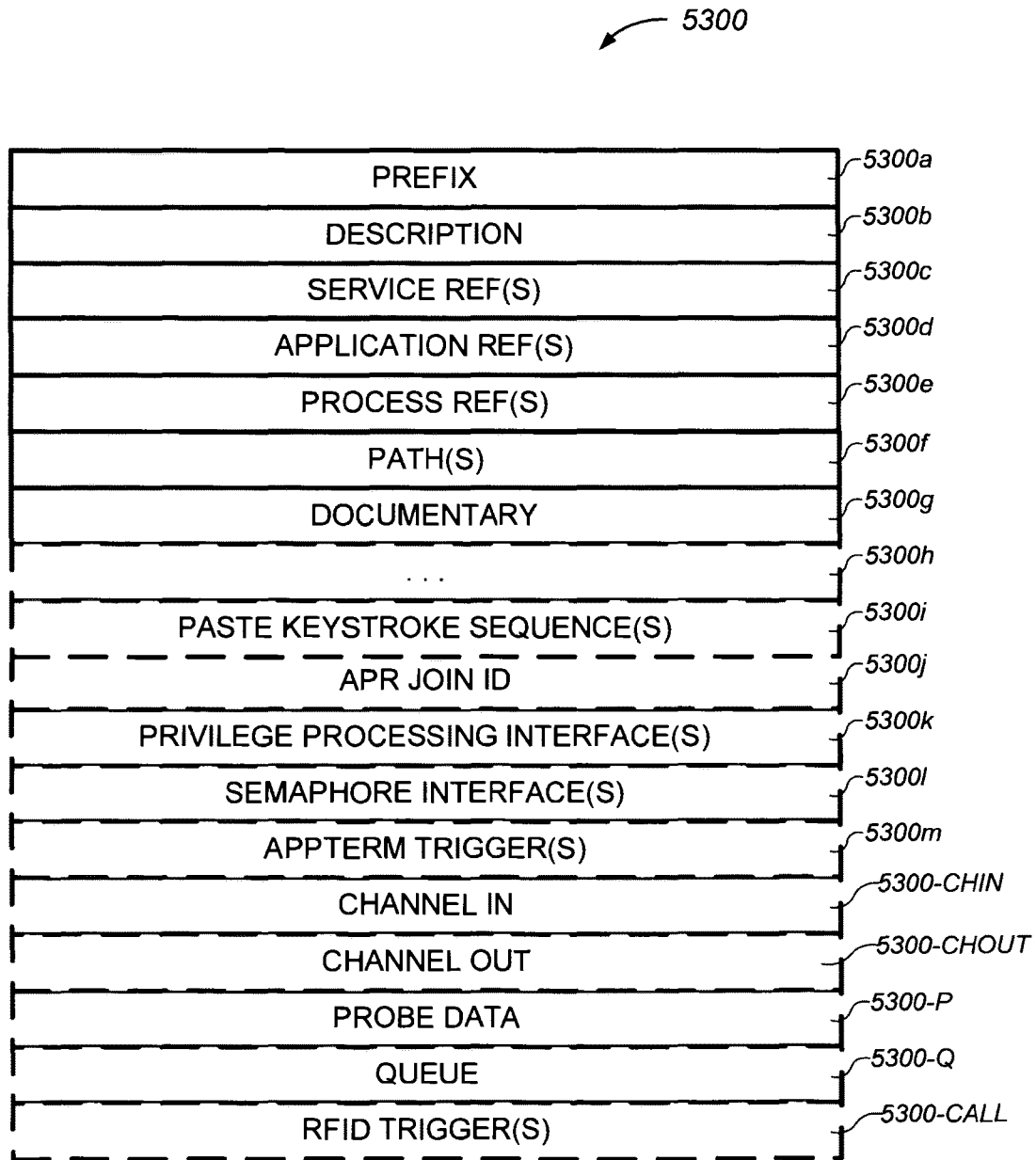
typedef struct permission {
    unsigned char    grantor[MAX_IDLENGTH];
    unsigned short   grantor_idtype;
    unsigned char    grantee[MAX_IDLENGTH];
    unsigned short   grantee_idtype;
    PRIVILEGE        *privileges;
    struct permission *nextPerm;
} PERMISSION;

typedef struct action {
    IDENTITY         host;
    unsigned short   cmd;
    unsigned short   operand;
    unsigned char    *params;
    TIMESPEC        *tspec;        // merged with charter level (if charter
                                    // level was present)
    struct action    *nextActn;
} ACTION;

typedef struct charter {
    unsigned char    grantee[MAX_IDLENGTH];
    unsigned short   grantee_idtype;
    unsigned char    grantor[MAX_IDLENGTH];
    unsigned short   grantor_idtype;
    unsigned char    *expression;
    ACTION           *actn;
    struct charter   *nextCharter;
} CHARTER;

```

Fig. 52

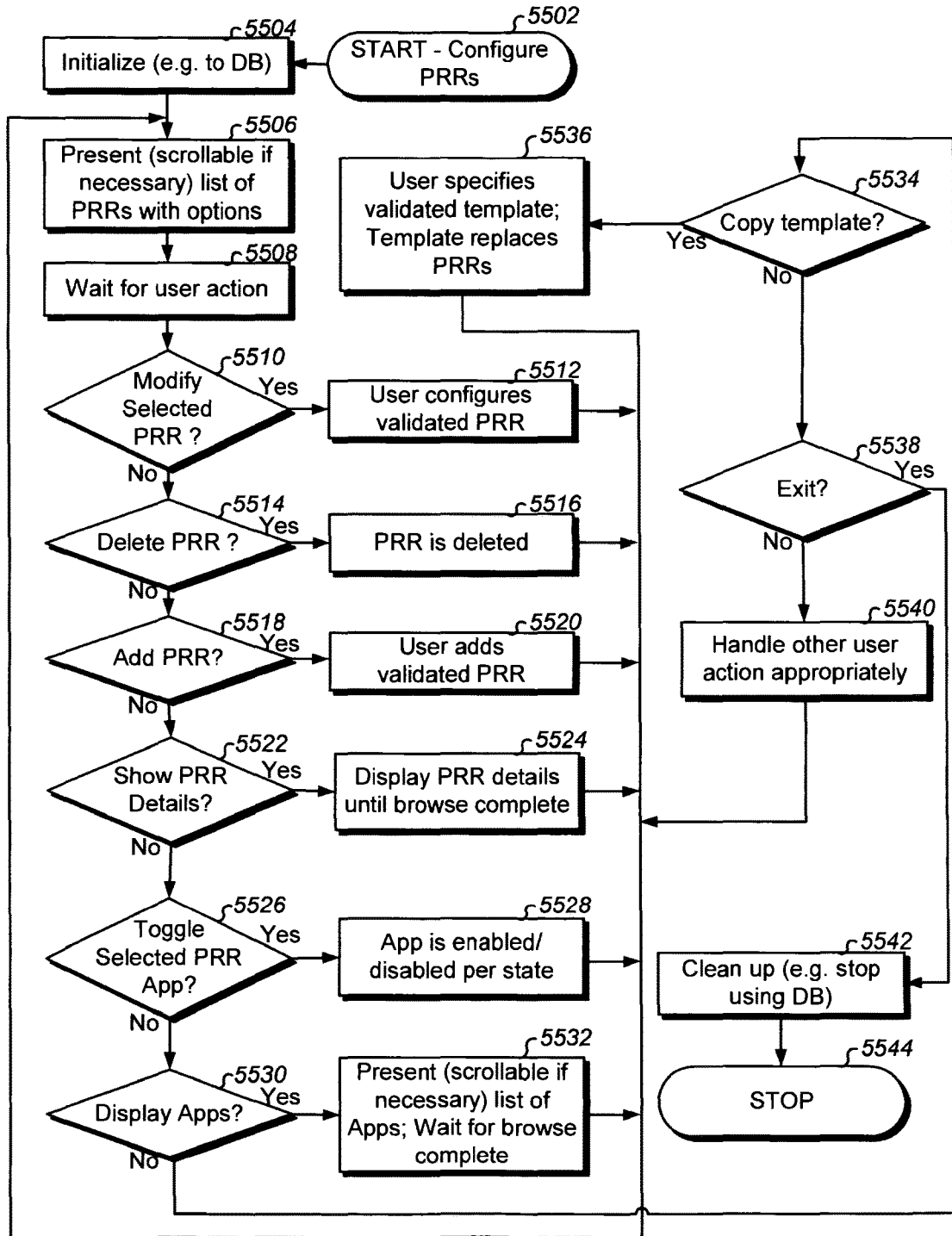
**Fig. 53**

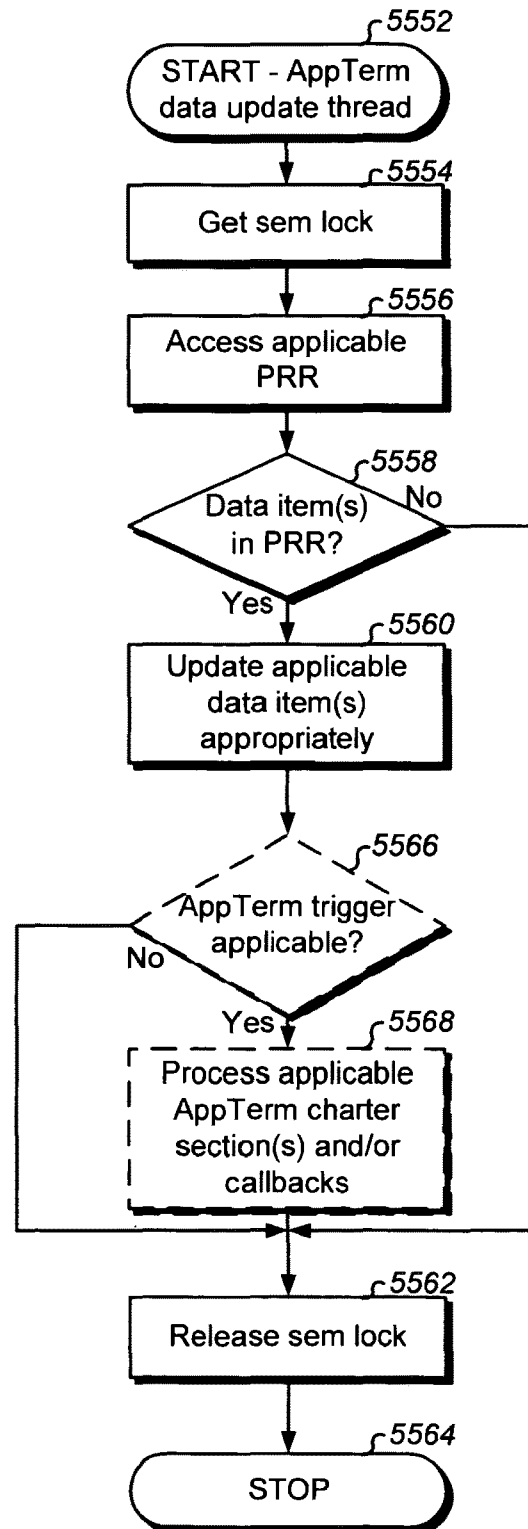
```

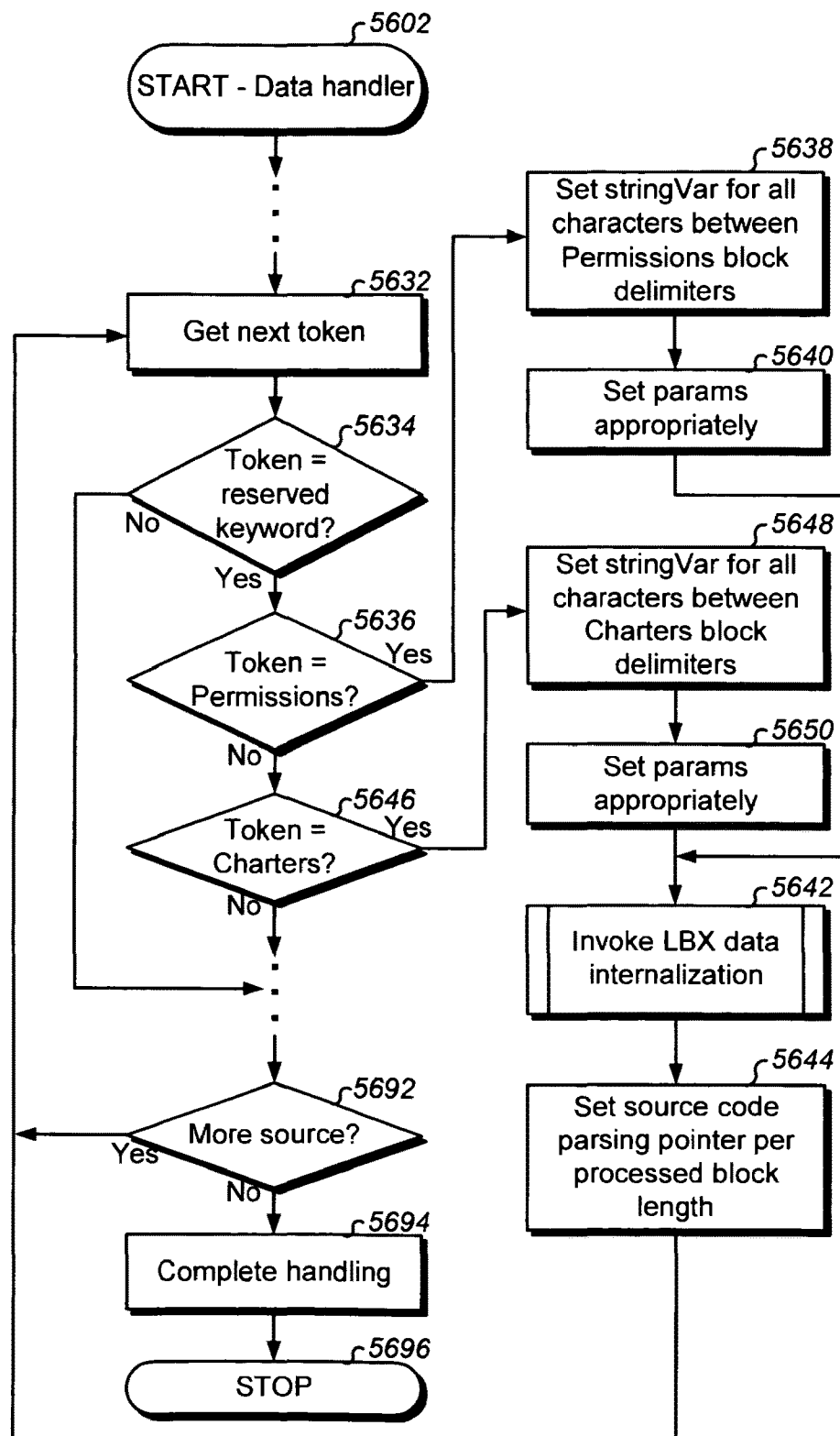
...
x = "this is a textual description"
...
z="timespec=""200802030000:200812312359"" description=""test98341;Permission""
...
<permission grantor="Jimbo" grantee="Henry" <%=z%> >
  <grant name="grant1" >
    <privilege id="\lbxcpy" relevance="FFFFFFF"
      timespec="YYMMDD09:YYMMDD17" description="<%=x%>" />
    <privilege id="\lbxflt" />
    ...
  </grant>
  ...
</permission>
...
<group name="group123" >
  <member="Jim" />
  <member="Sue" />
  ...
</group>
...
<charter grantee="Henry" grantor="Jimbo" timespec="200802030000:200812312359"
  description="test98341;Charter" >
  <expression>
    <condition trigger="true"
      specification="(__msid = ""Michael"") & __location $(300M) \loc_my" />
    <condition trigger="true"
      specification="(__msid = ""Jason"") & __location $(300M) \loc_my" />
  </expression>
  <action host="George" cmd="Invoke" operand="App" param="alert.cmd" />
  <action host="George" cmd="Notify" operand="Indicator" param="test98341 Fired!" />
</charter>
...

```

Fig. 54

**Fig. 55A**

**Fig. 55B**

**Fig. 56**

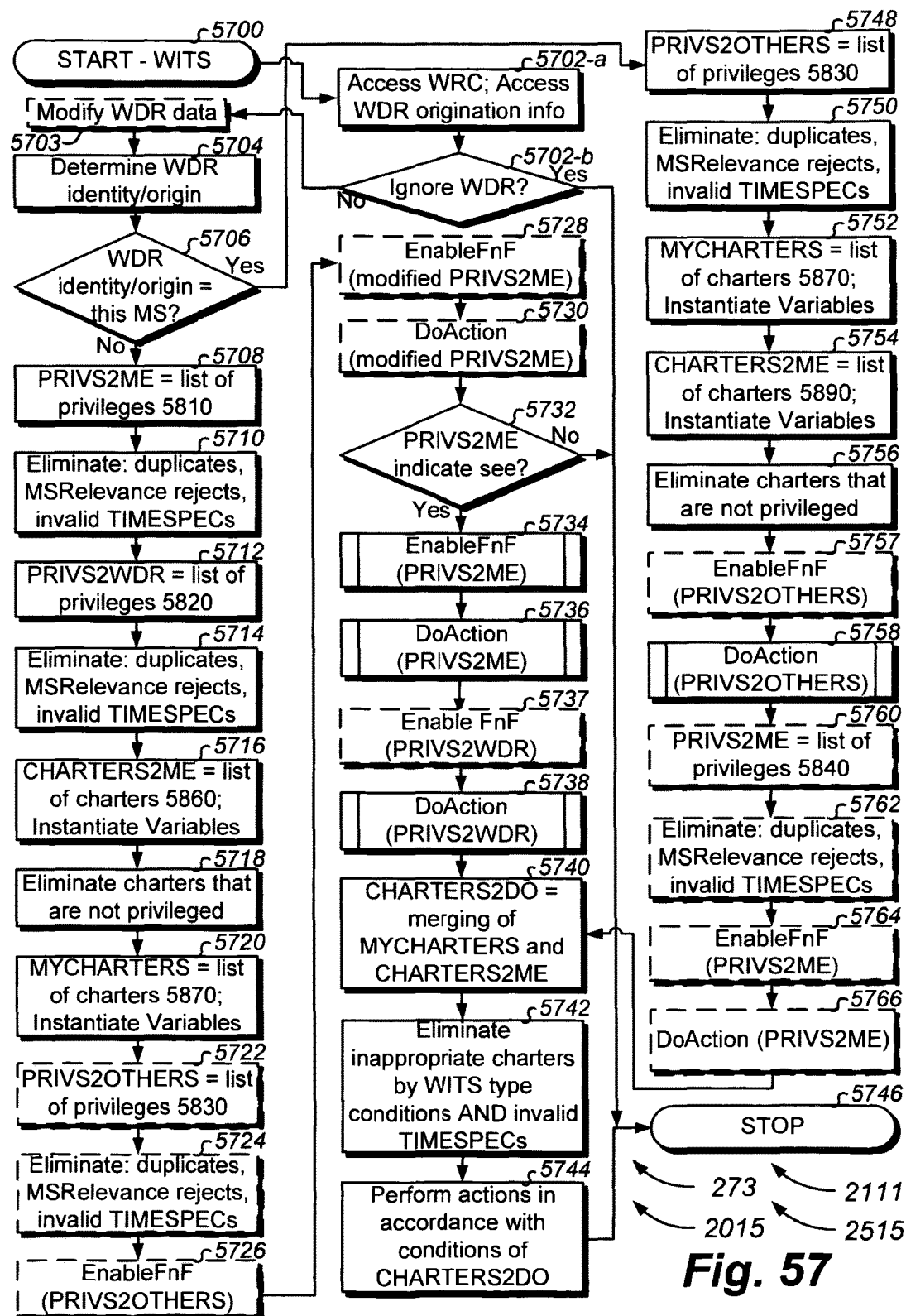
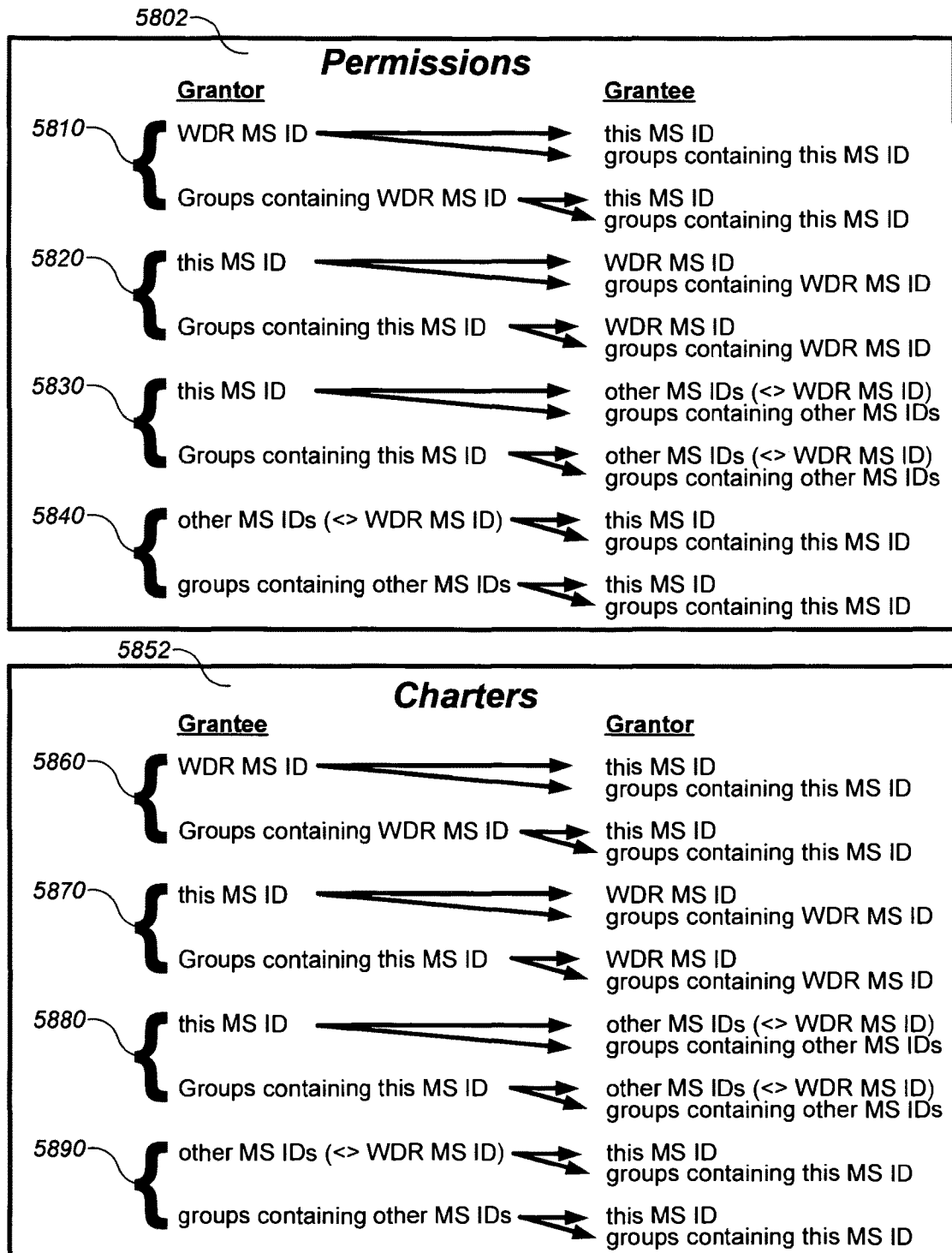
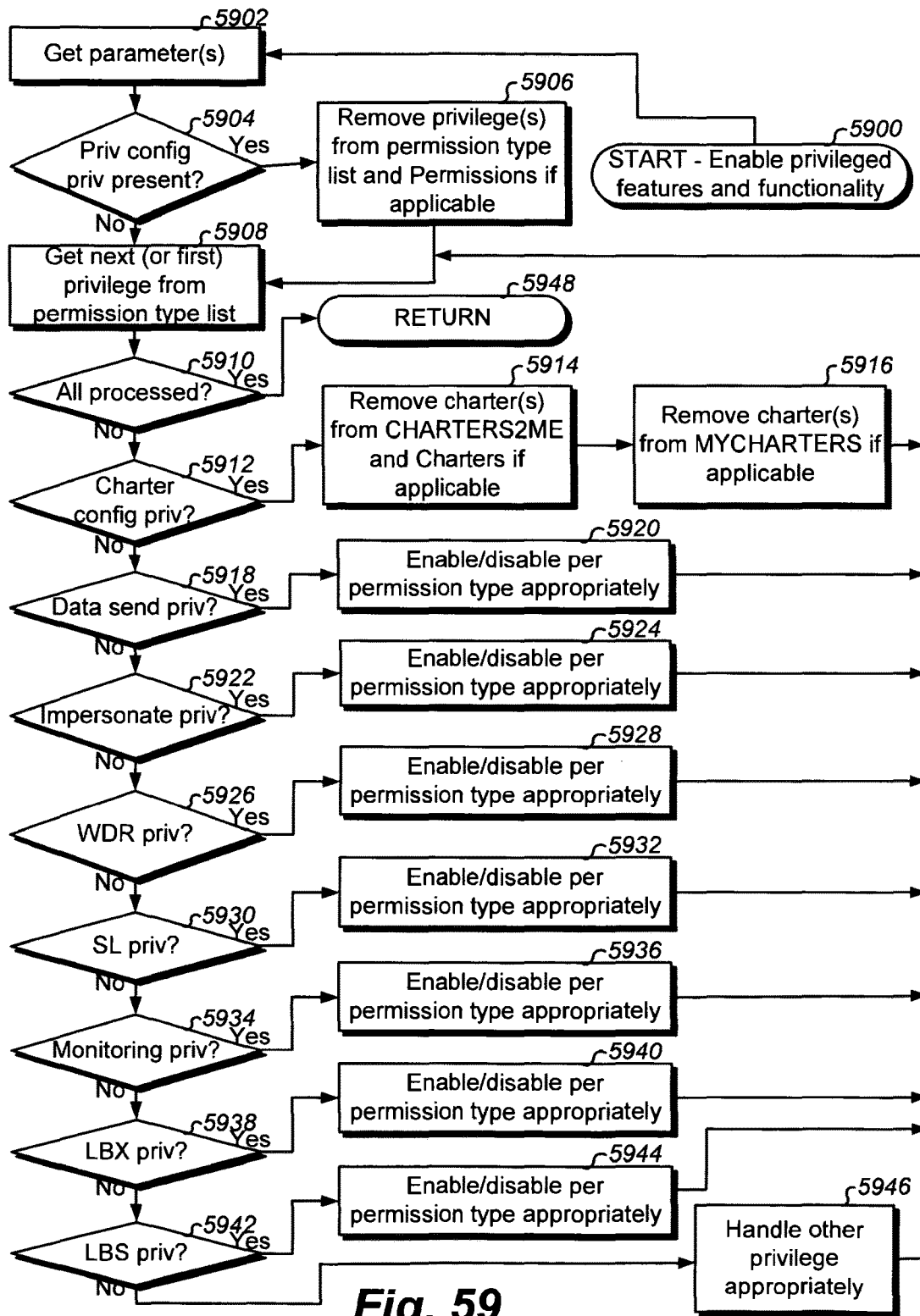
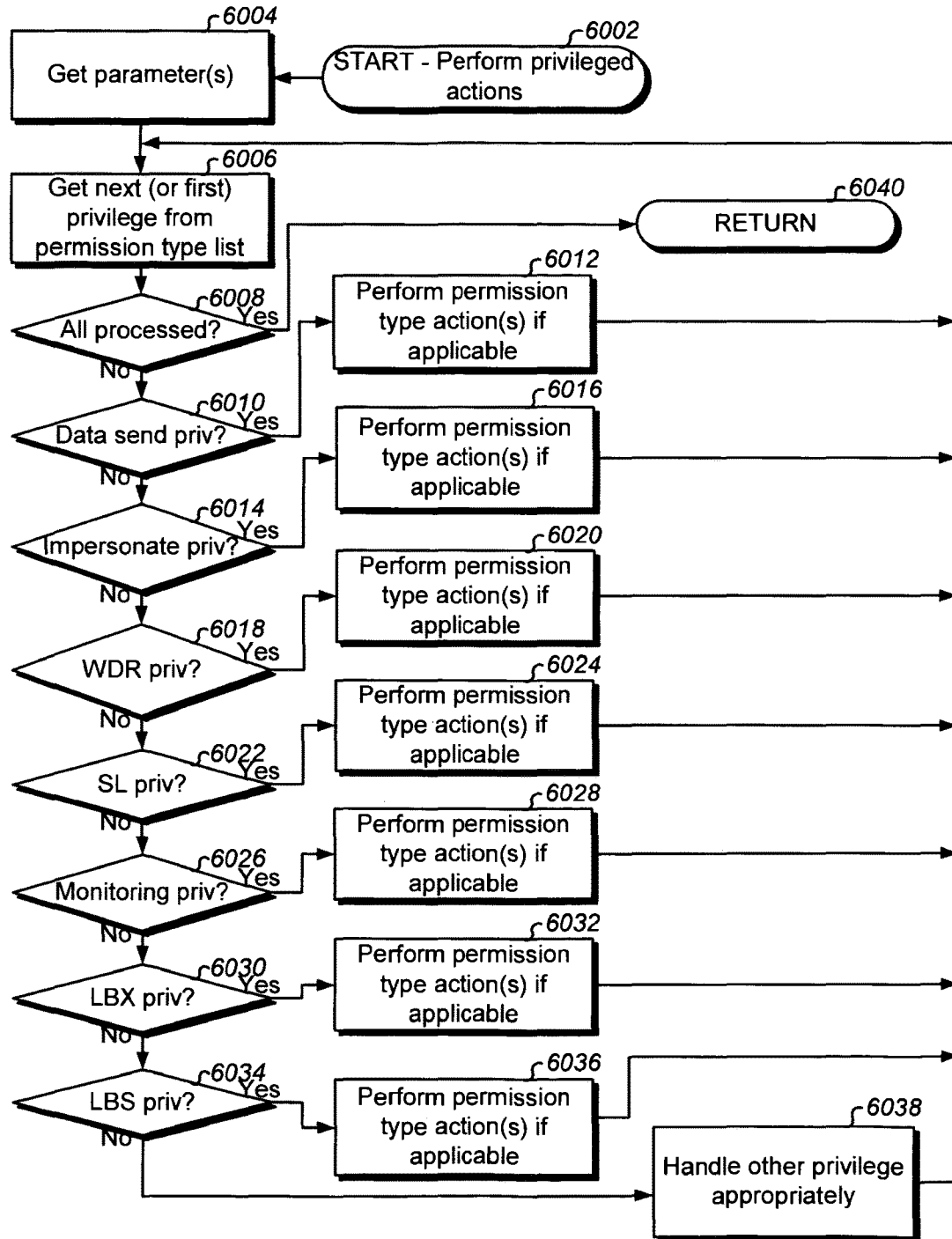


Fig. 57

**Fig. 58**



**Fig. 60**

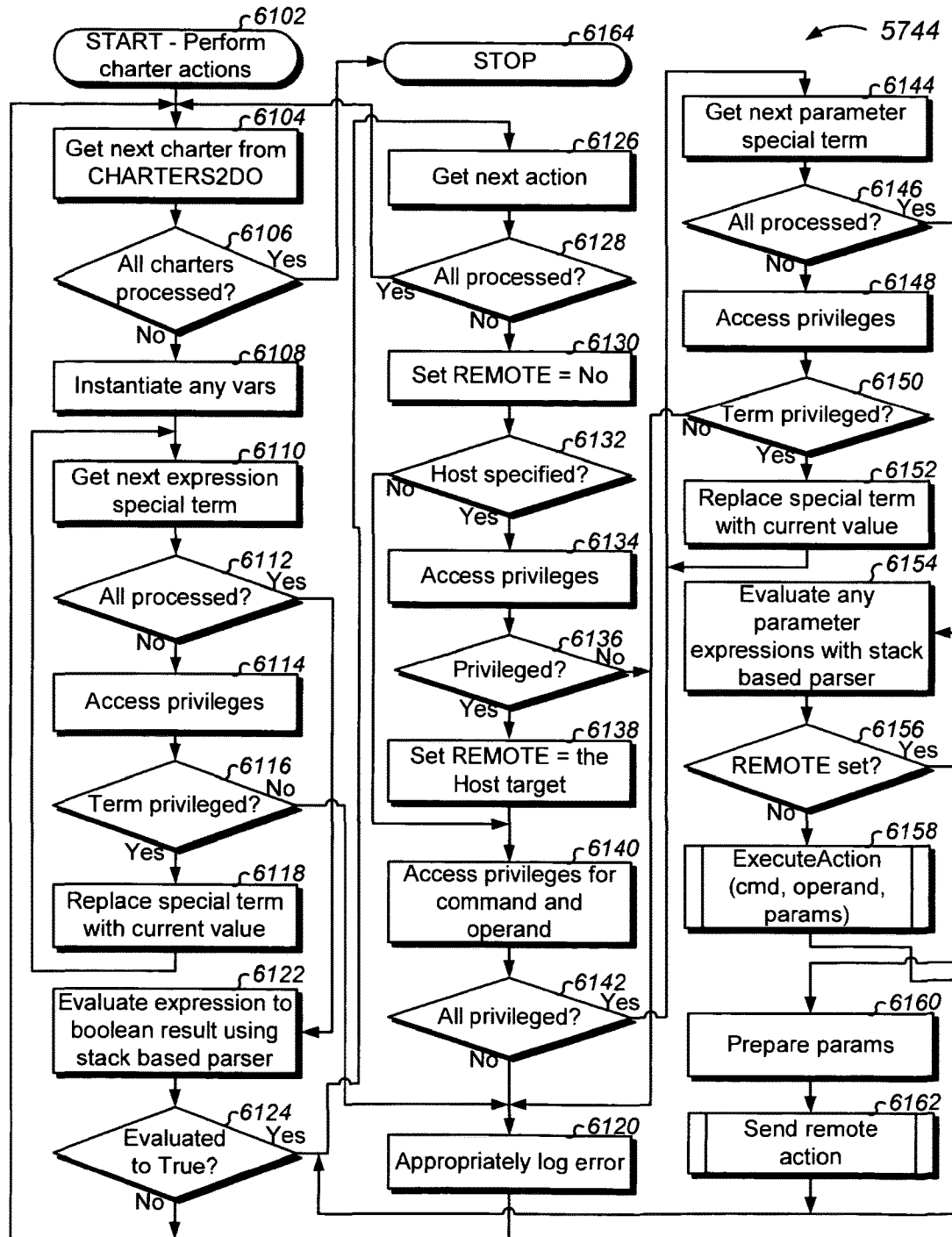
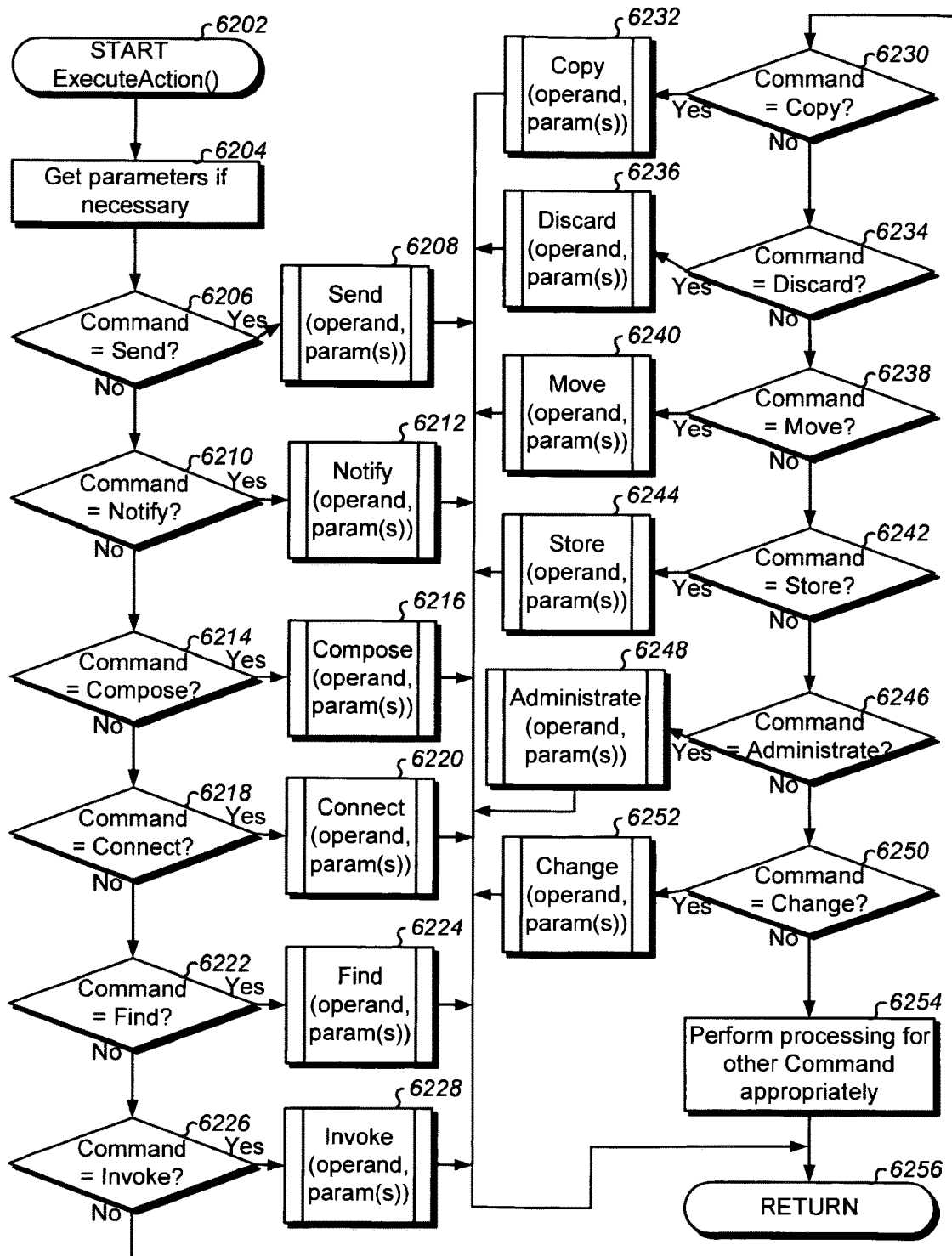
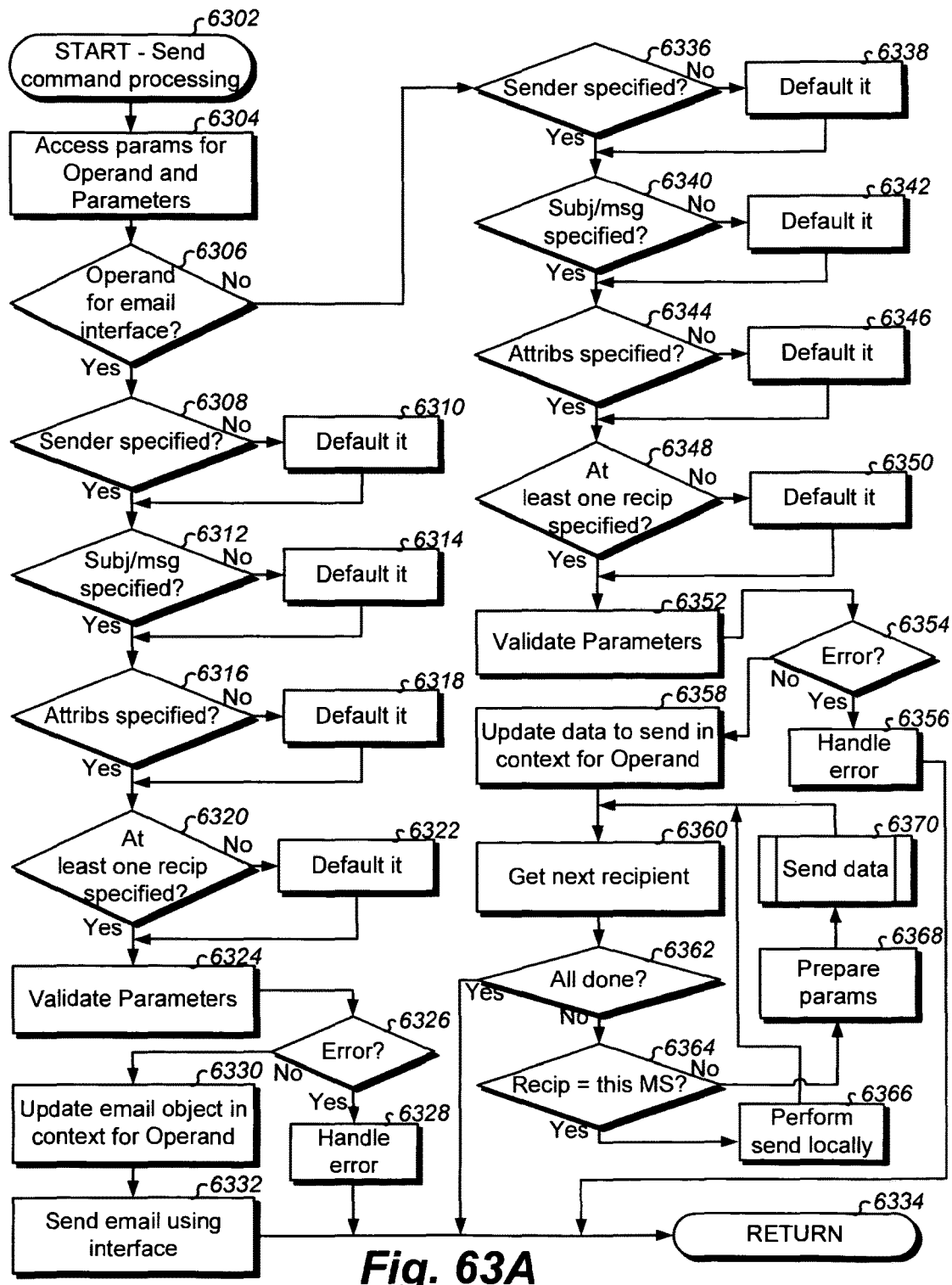


Fig. 61

**Fig. 62**



<u>Operand</u> ↓	<u>PM</u>	<u>Preferred embodiment Send processing</u>
201	O	Sending an auto-dial # updates appropriate recipient MS storage so that a recipient user can subsequently auto-dial the auto-dial # with a minimal user interface action. Preferably, the recipient MS user is appropriately and immediately notified of the receipt. Preferably, the send command data is maintained to LBX History, a historical call log (e.g. incoming), or other useful storage for subsequent user browse of the accompanying message and a date/time stamp of when sent, and for automated speed dialing of the # in response to a user action to auto-dial. Various embodiments will save to LBX History how many times, and when, the auto-dial # was used to perform automated speed dialing.
203	O	Sending a web link updates appropriate recipient MS storage so a recipient user can subsequently invoke (transpose to) the link, for example in a browser, with a minimal user interface action. Preferably, the recipient MS user is appropriately and immediately notified of the receipt. Preferably, the send command data is maintained to LBX History, a historical call log (e.g. incoming), browser history data, browser favorites, or other useful storage for subsequent user browse of the accompanying message and a date/time stamp of when sent, and for invocation of the link within a MS browser in response to a user action to use the link. Various embodiments will save to LBX History how many times, and when, the weblink was invoked.
205	E	Sending an email causes interface to the email delivery system (e.g. SMTP API) for sending the email body parameter. In one embodiment, the body is assumed to be the body of the email. In another embodiment, the body is attached with or without attachment(s). Attachments are preferably referenced with an appropriate syntax in the body specified. In another embodiment, the body is parsed for determining and using the best delivery options. The email will arrive to a recipient like other emails. Attributes can be set as is customary for email attributes (e.g. confirmation of delivery status, special handling, NLS considerations, etc).
207	E	Sending an SMS message causes interface to the sms message delivery system (e.g. SMTP API) for sending the sms message. The email interface can be used provided the sms message length maximum is observed. In one embodiment, the message parameter is identical to the msg/subj parameter. In another embodiment, the two parameters are concatenated, or formed in a complimentary manner, to highlight the subj/msg parameter from the sms message. In another embodiment, only a null subj/msg is supported. The message will arrive to a recipient like other sms messages. Various attributes can be set (e.g. confirmation of delivery status, special handling, NLS considerations, etc).

Fig. 63B-1

<i>Operand</i> ↓	PM	<u>Preferred embodiment Send processing</u>
209	E	Sending a broadcast email causes interface to the email delivery system (e.g. SMTP API) for sending the email body parameter. In one embodiment, the body is assumed to be the body of the email. In another embodiment, the body is attached with or without attachment(s). Attachments are preferably referenced with an appropriate syntax in the body specified. In another embodiment, the body is parsed for determining and using the best delivery options. The email will arrive to a recipient like other emails. Various attributes can be set (e.g. special handling, NLS considerations, etc), but preferably, no confirmation of delivery is set since this is a broadcast.
211	E	Sending an SMS broadcast message causes interface to the sms message delivery system (e.g. SMTP API) for sending the sms message parameter. The email interface can be used provided the sms message length maximum is observed. In one embodiment, the message parameter is identical to the msg/subj parameter. In another embodiment, the two parameters are concatenated, or formed in a complimentary manner, to highlight the subj/msg parameter from the sms message. In another embodiment, only a null subj/msg is supported. The message will arrive to a recipient like other messages. Various attributes can be set (e.g. special handling, NLS considerations, etc), but preferably, no confirmation of delivery status is set since this is a broadcast.
213	O	Sending an indicator updates appropriate recipient MS storage so that the currently focused user interface object (e.g. window titlebar) of the MS user interface is modified with the indicator. If there are no active user interface objects in the current MS user interface, then an appropriate alert area of the currently focused interface is to display the indicator. The user can clear (remove) the indicator when desired. Preferably, the indicator is used for modifying other focused objects (e.g. titlebars) or other focused areas in the user interface so as to not get overlooked. For example, as the user navigates and surfaces/focuses new user interface objects, the indicator remains visible on the newly focused object. Preferably, the indicator is selectable by the user of the MS for showing all other send command parameters associated, as well as a date/time stamp of when sent. In other embodiments, the most recently displayed indicator is displayed in the appropriate focused area, but the user can conveniently select any indicators which were sent in history at some point in time for sought indicator information by selecting the currently displayed indicator and then requesting to browse/scroll history of previously delivered indicators (with options to see details). Preferably, the send command data is maintained to LBX History, a historical log (web page load history), or other useful storage for subsequent use. Some title bar management methods include various IBM Technical Disclosure Bulletins from 1991 through 1995 (e.g. DA8-92-0910 "Originator Identified Direct Access Mail Basket Title Bar Mechanism", DA8-93-0061 "Roving Title Bar", DA8-93-0223 "Roving Title Bar Status", etc).

Fig. 63B-2

<i>Operand</i>	PM	<u>Preferred embodiment Send processing</u>
215	O	Sending an application causes invocation of the application at the recipient MS. The app parameter is preferably a fully qualified path name to the executable to start. In another embodiment, the app parameter is indirect: a path name to a "shortcut" (like a MS Windows shortcut). In another embodiment, the app parameter is an identifier string for the underlying operating system to know which application to start. The attributes parameter can be used for how to start the application, for example to flag whether to start an additional instance if the application is already running at the MS (provided multiple instances are supported). The msg/subject parameter may be useful for maintaining to LBX history useful information, along with a date/time stamp when sent, with record of the application invocation reference. An error is logged if the app parameter is not found for launch. Preferably, the invoke command data is maintained to LBX History, a historical log (web page load history), or other useful storage for subsequent use.
217	E	Sending a document causes interface to the email delivery system (e.g. SMTP API) for appropriately sending the document. The doc parameter is preferably a fully qualified path name, or suitable reference, to the document which may have a document type (e.g. by file extension, document parse, or document location). The document type is used for setting proper email attachment settings and perhaps the attributes parameter. Depending on the document type, the document may form the email body or be an attachment. The email will arrive to a recipient like other emails. Various attributes can be set (e.g. confirmation of delivery status, special handling, NLS considerations, etc).
219	E	Sending a file causes interface to the email delivery system (e.g. SMTP API) for appropriately sending the file. The path parameter is preferably a fully qualified path name, or suitable reference, to the file which should have a file type (e.g. by file extension, file parse, or file location). The file type is used for setting proper email attachment settings and perhaps the attributes parameter. Depending on the file type, the file may form the email body or be an attachment. The email will arrive to a recipient like other emails. Various attributes can be set (e.g. confirmation of delivery status, special handling, NLS considerations, etc).
221	O	Sending content causes the content to be sent to the recipient MS in a manner which is appropriate for where the content is stored and how it is to be subsequently presented. The content parameter is one that cannot be classified in the other operands, but is content for presentation nevertheless. Examples include special data records (e.g. extern variable name), content data memory locations (e.g. programmatic variable), or files containing a customizably processed format. Methods of displaying the content include audio and/or visual using applicable MS capabilities. Preferably, the send command data is maintained to LBX History, a historical content log, or other useful storage for subsequent user browse of the accompanying content and a date/time stamp of when sent, and for presentation of the content in response to an applicable user action. Attributes may be set for special content handling.

Fig. 63B-3

<i>Operand</i>	PM	<u>Preferred embodiment Send processing</u>
223	O	Sending a Database (DB) object causes the DB object to be sent to the recipient MS in a manner which is appropriate for subsequent import DB or table(s). The DB-obj parameter takes on many syntaxes for sending any subset of a database object, such as an entire database, table(s), certain rows, certain columns, etc. In one embodiment, a qualified database form is used such as: Owner:DatabaseName:TableName for sending the entire table (can use table name wildcard for multiple tables). In another embodiment, Owner:DatabaseName:"...SQL query..." shall return the data that is to be sent, preferably in a comma delimited or tab delimited form (as specified in the attributes parameter). Preferably, the send command data is maintained to LBX History, a database log, or other useful storage (subj/msg to document the transaction) for subsequent user browse of the accompanying data and a date/time stamp of when sent, and for DB query manager browse of the data in response to a user action for browse. One preferred embodiment enables the data for easy import to a variety of database destinations, preferably via the same DB query mgr interface(s) used for browsing.
225	O	Sending data causes reading the current value of the data at the MS where the send command action is being executed and then sending the current value to the recipient MS for informative purposes. In the preferred embodiment, the data is a global system variable visible to all processes of a MS operating system. In other embodiments, the data may have limited scope which is made accessible to present disclosure processing (e.g. with extern). An AppTerm uses record 5300 for access. Depending on the embodiment, data may be that which is contained in a program data segment, stack segment, and/or extra segment. There can be unique syntaxes for specifying which type of data is being sought (e.g. "S:dataname"). In the preferred embodiment, all occurrences found on the MS and information including its value about the occurrence is presented to the user. In one embodiment, a well known location of link symbol information files are consulted, and in another embodiment a new parameter specifies where to look, or which symbol file of information to use. Preferably, the data value is maintained to LBX History, a historical log, or other useful storage for subsequent user browse, or programmatic access, of the data variable name, its value and date/time stamp of when sent, and for presentation of the data value in response to a user action to show it.
227	O	Sending a semaphore causes reading the current value of the semaphore at the MS where the send command action is being executed and then sending the current value to the recipient MS for informative purposes. In the preferred embodiment, the semaphore is a global system semaphore visible to all processes of a MS operating system. In other embodiments, the semaphore may have limited scope which is made accessible to present disclosure processing (e.g. RAM semaphore). Preferably, the semaphore value (cleared or set) is maintained to LBX History, a historical log, or other useful storage for subsequent user browse, or programmatic access, of the semaphore name, its value and date/time stamp of when sent, and for presentation of the semaphore value in response to a user action to show it.

Fig. 63B-4

<i>Operand</i>	PM	Preferred embodiment Send processing
229	E	Sending a directory causes interface to the email delivery system (e.g. SMTP API) for sending the directory. In one embodiment, the directory is assumed to be the body of the email (e.g. when attributes parameter indicates it is to be a description only of the directory) for sending information about the directory such as # files, nesting of folders, sizes, and any useful file system characteristic(s) or statistics of the directory. In another embodiment, or as specified with an additional parameter (or in attributes), the directory is compressed and encoded as an attachment. In another embodiment, the directory is sent as individually attached files (as indicated to send that way by new or attributes parameter). The email will arrive to a recipient like other emails. The attribute parameter can be used for conventional email attributes as well as new attributes which affect directory data processing.
231	O	Sending an application context causes invocation of the application at the recipient MS and then executing a macro within the application context. The app parameter is preferably a fully qualified path name to the executable to start. In another embodiment, the app parameter is indirect: a path name to a "shortcut" (like a MS Windows shortcut). In another embodiment, the app parameter is an identifier string for the underlying operating system to know which application to start. The macro parameter is preferably a file, path, or accessible variable name containing a set of keystrokes that can be directed to standard/user-interface input. In another embodiment, the macro parameter is a prerecorded user input scenario (for play after application launched -- pulldown selections, mouse droppings, clicks, etc) captured to a file or stored in an accessible variable name. The attributes parameter can be used for how to start the application, for example to flag whether to start an additional instance if the application is already running at the MS (provided multiple instances are supported), and to specify the type of macro parameter being specified, or to specify a speed for processing individuals of the macro. The msg/subject parameter may be useful for maintaining to LBX history useful information with record of the application context invocation reference. An error is logged if the app parameter is not found for launch.
233	E	Sending the focused user interface object causes interface to the email delivery system (e.g. SMTP API) for sending an image (preferably .JPG) of the currently focused user interface object as an attachment. The "<alt><prtsrn>" constant string parameter is a syntactical string representation for the keystroke sequence for performing the MS focused user interface capture action. A similar syntax can be used to specify a different keystroke sequence (1 st parameter) for the same functionality. The email will arrive to a recipient like other emails. The attributes parameter can be set for which format to send, in which case a conversion may take place prior to sending (depends on embodiment). Various attributes can be set (e.g. confirmation of delivery status, special handling, NLS considerations, etc).

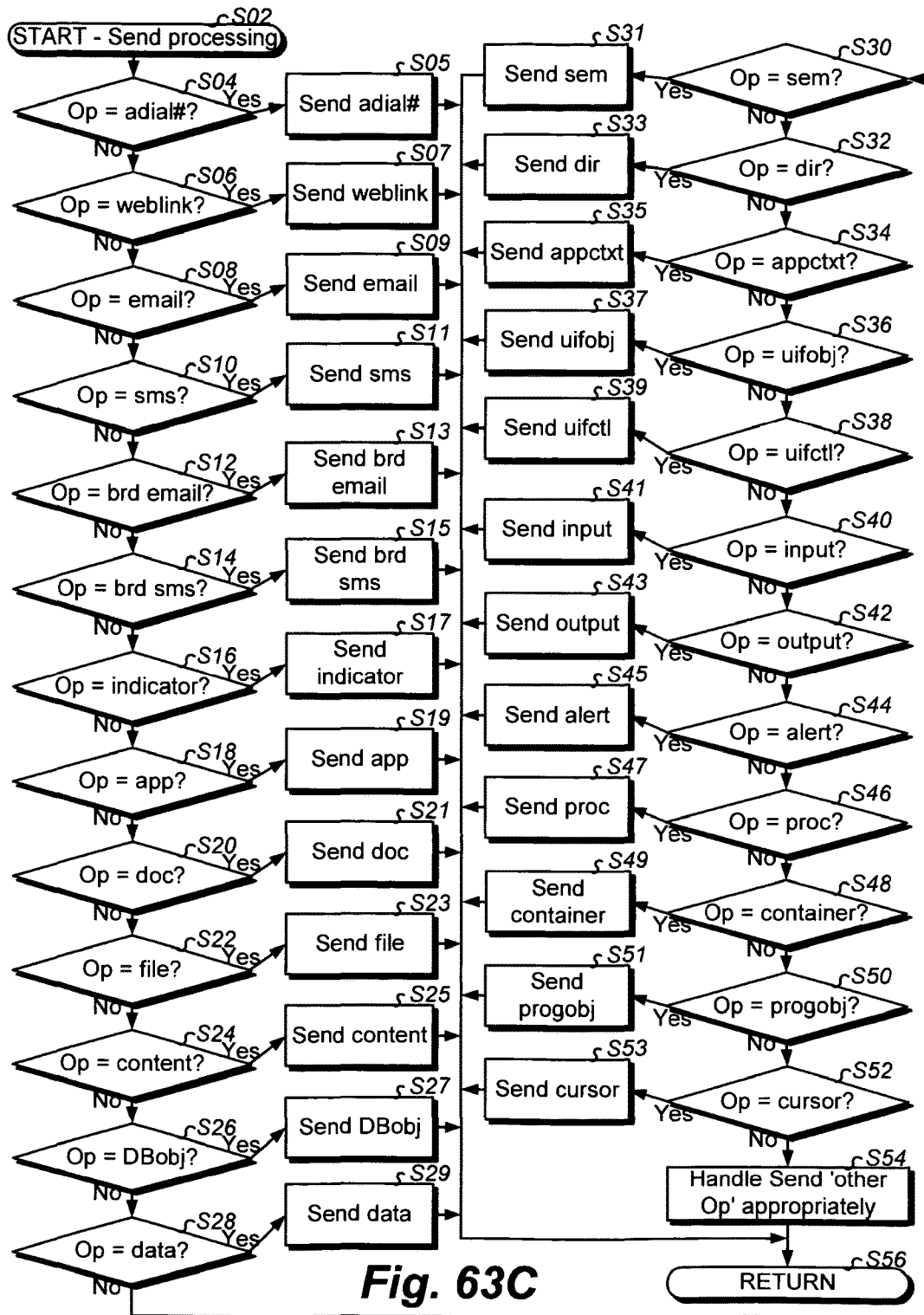
Fig. 63B-5

<i>Operand</i> ↓	PM	<u>Preferred embodiment Send processing</u>
235	O	Sending user interface control causes redirecting the keystroke macro to input of the recipient MS as if it were entered by the MS user. The macro parameter is preferably a file, path, or accessible variable name containing a set of keystrokes that can be directed to standard user-interface/input. In another embodiment, the macro is a prerecorded user input scenario (for play after application launched -- pulldown selections, mouse droppings, clicks, etc) captured to a file or stored in an accessible variable name. The attributes parameter can be used for whether or not to first display the subj/msg to the recipient MS user for user acknowledgement, or cancellation, prior to executing the macro at the MS. This allows the user time to get the MS user interface in a desirable state if necessary for running the macro, and to see information of the origination (i.e. Parameters). The msg/subject parameter may be useful for maintaining to LBX history information with a record of user interface control sent.
237	O	Sending input causes redirecting the input to the iodev parameter input device stream of the recipient MS as if it were entered by the MS user, or programmatically specified to the iodev I/O device parameter by a data processing system process. The input parameter is preferably a file, path, or accessible variable name containing a datastream (e.g. macro) recognizable by the iodev connected device. The attributes parameter can be used for whether or not to first display the subj/msg to the recipient MS user for user acknowledgement, or cancellation, prior to redirecting the input parameter datastream at the MS, or to specify a speed for processing individuals of the input. This allows the user time to get the MS user interface, and any iodev devices, in a desirable state if necessary for running the input, and to see information of the origination (i.e. Parameters). The msg/subject parameter may be useful for maintaining to LBX history information with a record of the user interface control having been sent.
239	O	Sending output causes redirecting the output to the iodev parameter output device stream of the recipient MS as if it were entered by the MS user, or programmatically specified to the iodev I/O device parameter by a data processing system process. The output parameter is preferably a file, path, or accessible variable name containing a datastream (e.g. macro) recognizable by the iodev connected device. The attributes parameter can be used for whether or not to first display the subj/msg to the recipient MS user for user acknowledgement prior to redirecting the output parameter datastream at the MS, or to specify a speed for processing individuals of the output. This allows the user time to get the MS user interface, and any iodev devices, in a desirable state if necessary for running the output, and to see information of the origination (i.e. Parameters). The msg/subject parameter may be useful for maintaining to LBX history information with a record of the user interface control having been sent.

Fig. 63B-6

<i>Operand</i> ↓	PM	Preferred embodiment Send processing
241	O	Sending an alert updates appropriate recipient MS storage so that a recipient MS alerter process can pick up the alert and then alert the user. There are a variety of alert processes, the most basic of which monitors incoming messages and posts them to the user in an alerting manner. In one embodiment, the alert parameter is identical to the msg/subj parameter. In another embodiment, the two parameters are concatenated, or formed in a complimentary manner, to highlight the subj/msg parameter from the alert message. In another embodiment, only a null subj/msg is supported. The attributes parameter can be for special treatment of the alert by an alerter process.
243	O	See Notify Command for identical processing.
245	O	See Notify Command for identical processing.
247	O	See Notify Command for identical processing.
249	O	See Notify Command for identical processing.
251	O	Sending a calendar object causes interface to the recipient MS calendar/scheduling system for sending/scheduling the calendar object parameter. The calobj parameter contains the date/time stamp of when to schedule the object, or a special syntax constant for "now", "first available per recipient and sender availability", "by end of the week pending availability", or other reasonable constants for when to schedule the calendar object. In one embodiment, the calendar object is assumed to be a newly scheduled calendar item for placement to the calendar of recipients. In another embodiment, the calendar object (e.g. data or file containing parsable syntax) contains directives for what actions exactly to perform to the calendar application interface. In another embodiment, the email system is the transport to deliver the calendar object or calendar actions to recipients. Attributes can be set as is customary for calendar entries (attendance required, emergency meeting, recurring/weekly/monthly meeting, etc). The attributes parameter may be used for performing other actions/functions in the calendaring interface.
253	O	Sending an address book (AB) object causes interface to the AB system for sending/entering the AB object parameter at the recipient MS. In one embodiment, the AB object is assumed to be a newly entered AB entry (e.g. contact reference name) for creation in the AB of recipients. In another embodiment, the AB object parameter contains directives (e.g. data or file containing parsable syntax) for what actions exactly to perform to the AB application interface. Attributes can be set as may be customary for AB entries (customer, peer, manager, friend, family, etc). The attributes parameter may be used for performing other actions/functions in the AB interface.
...		

Fig. 63B-7



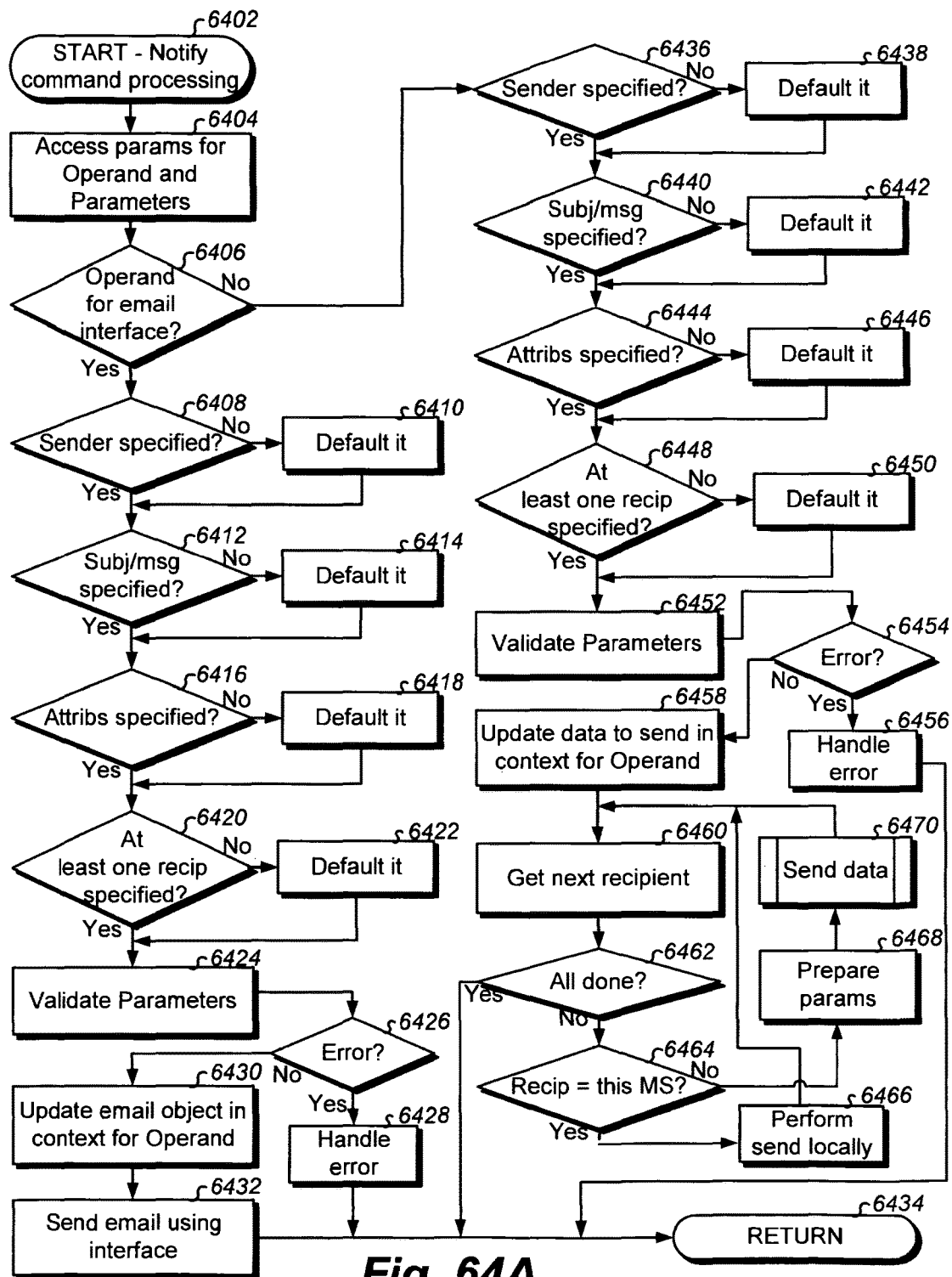


Fig. 64A

<i>Operand</i> ↓	PM	Preferred embodiment Notify processing
201	O	Notifying with an auto-dial # automatically performs call processing to auto-dial the auto-dial #. Preferably, the recipient MS user is called by the MS as a normal phone call would be made. In one embodiment, multiple recipients are called "back to back" after the previous recipient call terminates. In another embodiment, a multiple line party call is made with an automated manner with all recipients. The attributes parameter can indicate which embodiment to use, and can be used for specialized call processing (collect, prepaid account check, hide caller id, etc). Preferably, the notify command data and call data is maintained to LBX History, a historical call log (e.g. incoming), with the accompanying subj/msg and a date/time stamp of when sent, and for future repeated automated speed dialing of the # in response to a user action to auto-dial. Various embodiments will save to LBX History how many times, and when, the auto-dial # was used to perform automated speed dialing, along with call details such as direction of call, parties to the call, features of the call, or other call characteristics. In one embodiment, the recipient ID is one to one with the called #. In another embodiment, the recipient ID is used to find the associated called number. Preferably, an existing API is used to accomplish processing. Automatic dialing through a variety of interfaces is well known in the art, and depends on the software development environment. Conventional processing side affects of automated calling should occur like the action was manual (e.g. log update).
203	O	Notifying with a web link automatically invokes (transposes to) at the recipient MS the link, for example in a browser, with a minimal (if any) user interface action. In one embodiment, the link includes URL parameter(s) (e.g. ?p1=xyz). In another embodiment, all recipients are passed to the link with appended URL parameter(s) (e.g. ?ids=Recip1;Recip2;...RecipN). An alternate embodiment fires form variables to the loaded page with the same URL variables. The attributes parameter can indicate which embodiment to use, and how to invoke the link (e.g. use currently focused window, use an active browser window, spawn new browser window, etc). Preferably, the notify command data is maintained to LBX History, a historical call log (e.g. incoming), browser history data, browser favorites, or other useful storage for subsequent user browse of the accompanying subj/msg and a date/time stamp of when sent, and for invocation of the link within a MS browser in response to a user action to use the link again in the future. Various embodiments will save to LBX History how many times, and when, the weblink was invoked.
205	E	See Send Command for identical processing.
207	E	See Send Command for identical processing.
209	E	See Send Command for identical processing.
211	E	See Send Command for identical processing.
213	O	See Send Command for identical processing.
215	O	See Send Command for identical processing.

Fig. 64B-1

<i>Operand</i> ↓	PM	<u>Preferred embodiment Notify processing</u>
217	E	See Send Command for identical processing.
219	E	See Send Command for identical processing.
221	O	Notifying with content causes the content to be presented at the recipient MS upon delivery in a manner which is appropriate for the content type. The content parameter is one that cannot be classified in the other operands, but is content for presentation nevertheless. Examples include special data records (e.g. extern variable name), content data memory locations (e.g. programmatic variable), or files containing a customizably processed format. Methods of displaying the content include audio and/or visual using applicable MS capabilities. Preferably, the notify command data is maintained to LBX History, a historical content log (e.g. incoming), browser history data, or other useful storage for subsequent user browse of the accompanying content and a date/time stamp of when sent, and for presentation of the content. Various embodiments will save to LBX History how many times, and when, the content was presented.
223	O	Notifying with a Database (DB) object causes the DB object (i.e. qualified database with access query string) to be modified with the query parameter. The query parameter is used to perform any query against the specified DB-database (DB-obj), preferably a query that only returns a return code (e.g. causes alteration). Preferably, the notify command data is maintained to LBX History, a database log (e.g. incoming), or other useful storage for subsequent query use and a date/time stamp of when sent, and for DB query manager browse/use of the query in response to an applicable user action. Other params are for documentary purposes when information is saved. In some embodiments, an appropriate SQL client interface (e.g. SQLNET API) is used to carry out processing, or a suitable DB API is used.
225	O	Notifying data causes modifying the value of the data at the recipient MS (set data to value). An error can result if the data is not resolvable for the attempt. In the preferred embodiment, the data is a global system variable visible to all processes of a MS operating system. In other embodiments, the data may have limited scope which is made accessible to present disclosure processing (e.g. with extern). An AppTerm uses record 5300 for access. Preferably, the data affected is maintained to LBX History, a historical log (e.g. incoming), or other useful storage for subsequent user browse, or programmatic access, of the data variable name, its before and after values and date/time stamp of when sent, and for presentation of the data value in response to a user action to show it.
227	O	Notifying a semaphore causes modifying the value of the semaphore at the recipient MS. In the preferred embodiment, the semaphore is a global system semaphore visible to all processes of a MS operating system. In other embodiments, the semaphore may have limited scope which is made accessible to present disclosure processing (e.g. RAM semaphore). Preferably, the semaphore value before and after setting is maintained to LBX History, a historical log (e.g. incoming), or other useful storage for subsequent user browse, or programmatic access, and for presentation of the semaphore information in response to a user action to show it.

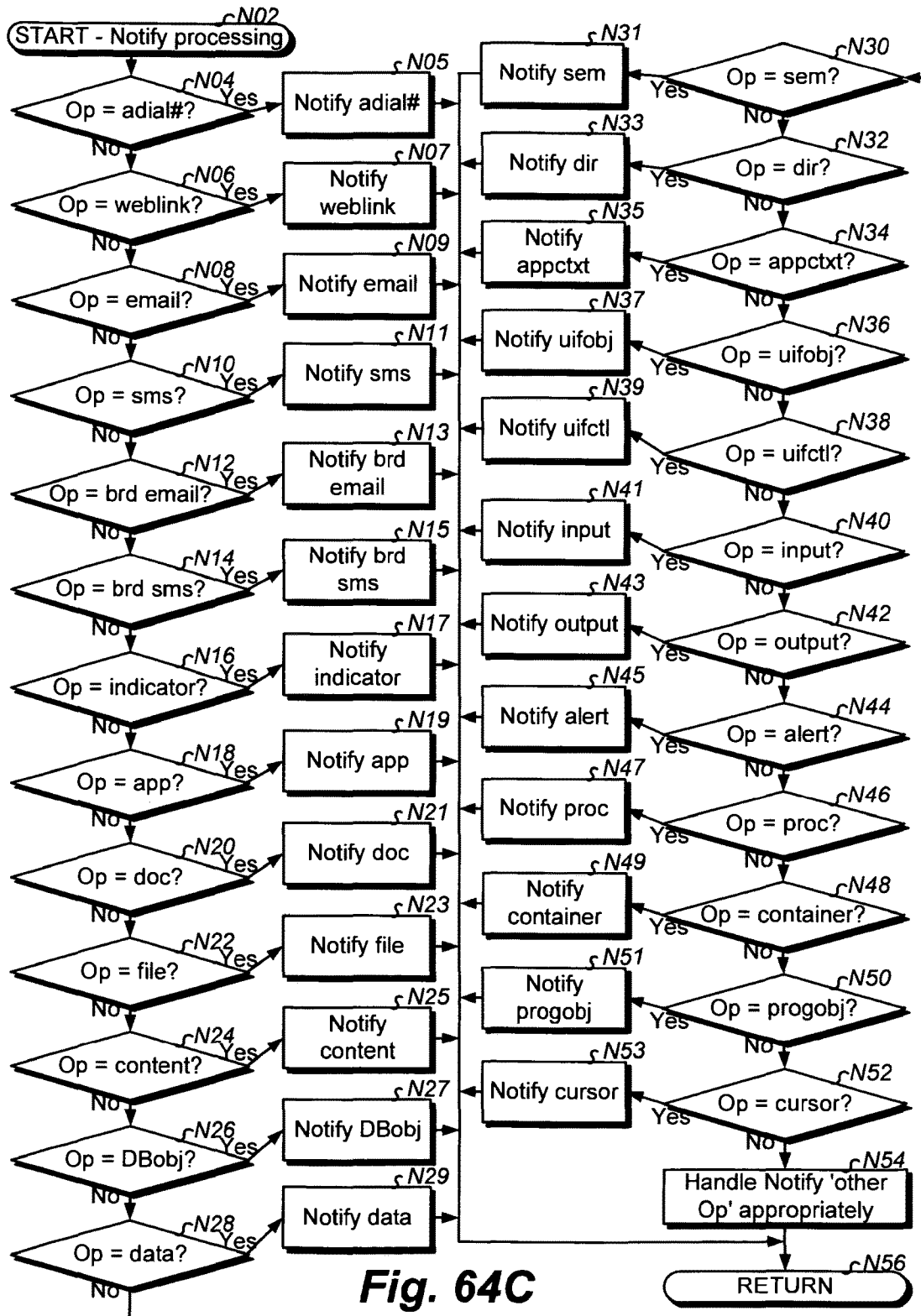
Fig. 64B-2

<i>Operand</i> ↓	PM	<u>Preferred embodiment Notify processing</u>
229	E	See Send Command for identical processing.
231	O	See Send Command for identical processing.
233	E	See Send Command for identical processing.
235	O	See Send Command for identical processing.
237	O	See Send Command for identical processing.
239	O	See Send Command for identical processing.
241	O	Notifying with an alert presents the alert to each recipient MS. Depending on attributes parameter settings, the alert may be asynchronously presented to an alert area, synchronously alerted and requiring a user action to acknowledge, logged to special file, or other reasonable alert method(s). Fig. 75B processing will cause the alert to be presented to the MS user. In one embodiment, the alert parameter is identical to the msg/subj parameter. In another embodiment, the two parameters are concatenated, or formed in a complimentary manner, to highlight the subj/msg parameter from the alert message. In another embodiment, only a null subj/msg is supported. Various embodiments will support different alert content types and applicable processing as indicated by the attributes parameter. Preferably, an appropriate API is made available for processing.
243	O	Notifying a process causes sending an operating system signal (see UNIX signaling) to the process with Process ID (PID) of the pid parameter. A numeric value parameter (e.g. 0 or 1) may be communicated with the signal. Depending on attributes parameter settings, another embodiment accesses the pid parameter as a process identifier parameter which is used to lookup the operating system PID prior to signaling. The msg/subject parameter may be useful for maintaining to LBX history useful information, along with a date/time stamp when sent, with record of the application invocation reference. An error can be logged if the process is not found for signaling.
245	O	Notifying a container causes launch of a MS file manager to examine the contents of a MS system container having the path in the container parameter (e.g. c:\dir1\subdir3). The attributes parameter can be used for how to start the file manager, for example to flag whether to start an additional instance if the file manager is already running at the MS (provided multiple instances are supported), otherwise an existing instance is updated for the container, or a new instance is started for the container. The msg/subject parameter may be useful for maintaining to LBX history useful information, along with a sent date/time stamp, with record of the application invocation reference. An error can be logged if the file manager is not found for launch, or if the container is invalid.

Fig. 64B-3

<i>Operand</i> ↓	PM	<u>Preferred embodiment Notify processing</u>
247	O	Notifying a program object causes acting on a specified program object (per attribute parameter) with the specified data at the recipient MS. The progobj parameter is the linked run time symbolic name accessible to charter processing of the present disclosure for third party plug-in processing. The progobj parameter can be a variable name, function name, object name, queue name, procedure name, or semaphore name accessed at run time by the symbolic name evaluation during charter processing. The binary data parameter is used to modify the program object (variable name set Least Significant Bit (LSB) to Most significant Bit (MSB) right to left intuitive Motorola processing byte/bit order until bits set or unmatched, function name invoked with respective data bytes pushed to the stack prior to invocation, object name data public data area initialized with the data parameter on a byte to byte basis, queue name entry inserted using the data parameter as a typecast data record of bits, procedure name invoked with respective data bytes pushed to the stack prior to invocation, or semaphore name set with clear for a null data parameter, else a set action). Alternately, an Intel reverse byte order can be used to apply the data Parameter. The attributes parameter indicates which variety of progobj is specified, and can be used to indicate a byte order data mapping method to use. The msg/subject parameter may be useful for maintaining to LBX history useful information, along with a date/time stamp when sent, with record of the program object invocation. An error can be logged if the progobj parameter is not resolvable. Appropriate MS O/S interfaces are used.
249	O	Notifying a cursor causes modifying a recipient MS user interface cursor in accordance with direction by the attributes parameter. The cursor parameter can be a suitable cursor bitmap file reference, suitable animated cursor file, predefined appearance type, or predefined behaving cursor. The attributes parameter further distinguishes which cursor modification is being requested. The msg/subject parameter may be useful for maintaining to LBX history useful information, along with a date/time stamp when sent. An error is logged if there is no active cursor in the user interface. An appropriate MS API is used, depending on the development environment.
251	O	See Send Command for identical processing.
253	O	See Send Command for identical processing.
...		

Fig. 64B-4



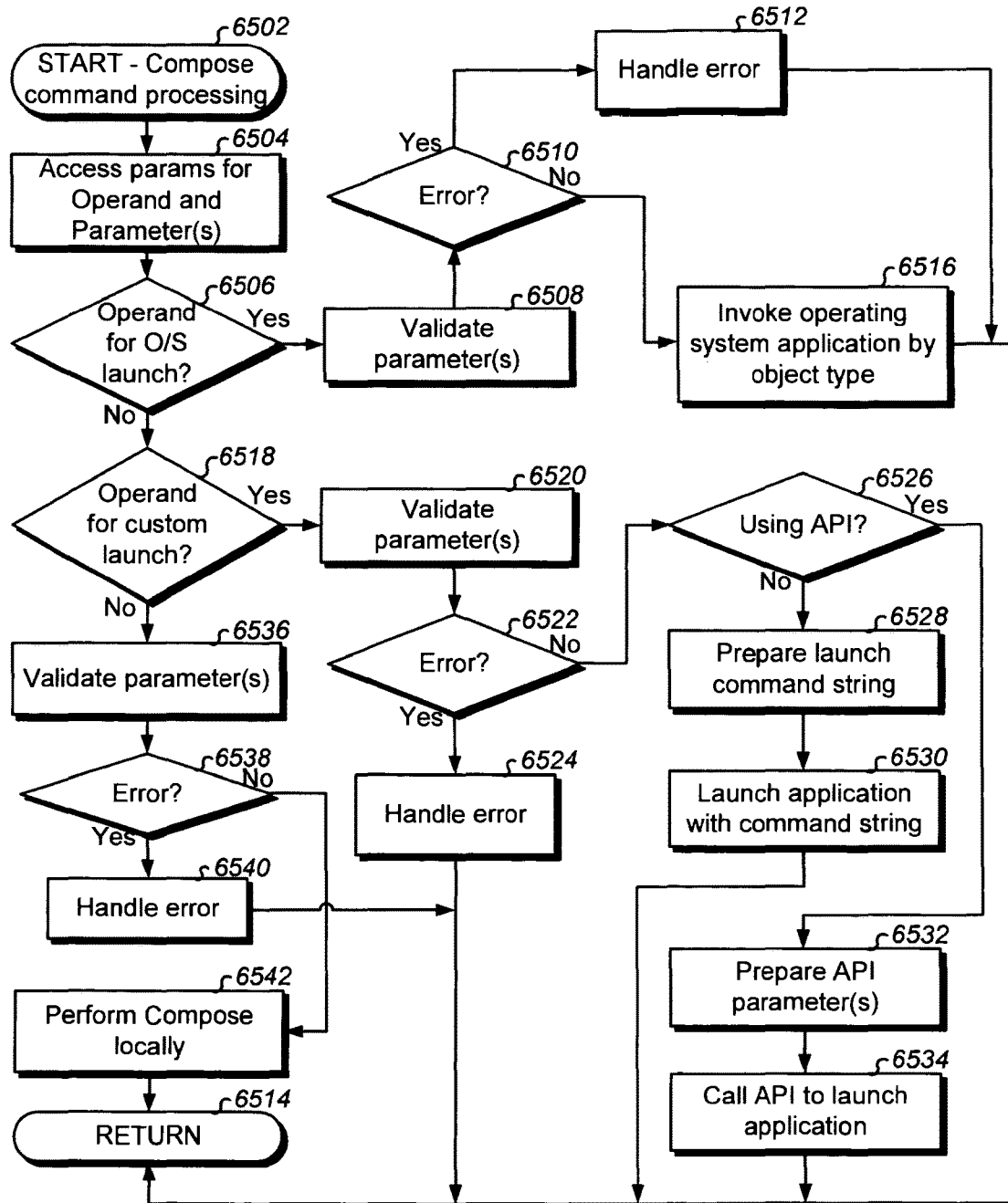


Fig. 65A

<i>Operand</i>	PM	<u>Preferred embodiment Compose processing</u>
201	C	Composing an auto-dial # launches the MS phone number calling interface with the auto-dial # parameter defaulted for making the call. Once launched, the user can make a very simple confirmation action for placing the call to the auto-dial #. Call processing takes place as though the user manually launched the dialing application, entered the auto-dial # and then is ready to decide if the call should be placed. Appropriate MS storage is updated and subsequently processed as though the user had entered the # for calling manually. Preferably, the compose command data is maintained to LBX History, a historical call log (e.g. outgoing when call placed), or other useful storage for subsequent use.
203	S	Composing a web link launches a MS browser and defaults the link as though the user had entered it manually. The user can subsequently invoke (transpose to) the link if desired with a minimal action (e.g. click ok). The link may include appended URL parameters (e.g. "?v=yes&T=go" for customized web page processing). An alternate embodiment can fire form variables for active web page processing using URL parameters specified or using the params parameter. Processing takes place as though the user manually launched the browser application, entered the weblink and then is ready to decide if the link should be invoked. Appropriate MS storage is updated and subsequently processed as though the user had entered the weblink in the browser manually. Preferably, the compose command data is maintained to LBX History, a historical log (web page load history when invoked), or other useful storage for subsequent use.
205	C	Composing an email causes interface to the email delivery system for invoking the create email interface and defaulting the appropriate email fields with the passed parameters. The user can subsequently send the email with little effort, or after optional modification, with a minimal action (e.g. click ok). Processing takes place as though the user manually launched the create email application, entered the fields of the email form with passed parameters, and then is ready to decide if the email should be sent, or further edited, or possibly cancelled. Appropriate MS storage is updated and subsequently processed as though the user manually started the create email interface and entered the email information manually. Preferably, the compose command data is maintained to LBX History, a historical call log (e.g. incoming), or other useful storage for subsequent use. A standard email POP or mailbox interface is preferably used. The email will arrive to a recipient like other emails. Various attributes can be set (e.g. confirmation of delivery status, special handling, NLS considerations, etc).

Fig. 65B-1

<i>Operand</i> ↓	PM	<u>Preferred embodiment Compose processing</u>
207	C	Composing an sms message causes interface to an appropriate messaging delivery system for invoking the create message interface and defaulting the appropriate message fields with the passed parameters. The user can subsequently send the message with little effort, or after optional modification, with a minimal action (e.g. click ok). Processing takes place as though the user manually launched the create message application, entered the fields of the messaging form with passed parameters, and then is ready to decide if the message should be sent, or further edited, or possibly cancelled. Appropriate MS storage is updated and subsequently processed as though the user manually started the create message interface and entered message information manually. Preferably, the compose command data is maintained to LBX History, a historical call log (e.g. incoming), or other useful storage for subsequent use. A standard email POP or mailbox interface, or a similar messaging interface, can be used. The message will arrive to a recipient like other sms messages. Various sms message attributes may be set (e.g. confirmation of delivery status, special handling, NLS considerations, etc).
209	C	Composing a broadcast email causes interface to the email delivery system for invoking the create email interface and defaulting the appropriate email fields with the passed parameters. The user can subsequently send the email with little effort, or after optional modification, with a minimal action (e.g. click ok). Processing takes place as though the user manually launched the create email application, entered the fields of the email form with passed parameters, and then is ready to decide if the email should be sent, or further edited, or possibly cancelled. Appropriate MS storage is updated and subsequently processed as though the user manually started the create email interface and entered the email information manually. Preferably, the compose command data is maintained to LBX History, a historical call log (e.g. incoming), or other useful storage for subsequent use. A standard email POP or mailbox interface is preferably used. The email will arrive to a recipient like other emails. Various attributes can be set (e.g. special handling, NLS considerations, etc), but preferably, no confirmation of delivery status is requested in attributes since this is a broadcast.

Fig. 65B-2

<i>Operand</i> ↓	PM	<u>Preferred embodiment Compose processing</u>
211	C	Composing a broadcast sms message causes interface to an appropriate messaging delivery system for invoking the create message interface and defaulting the appropriate message fields with the passed parameters. The user can subsequently send the message with little effort, or after optional modification, with a minimal action (e.g. click ok). Processing takes place as though the user manually launched the create message application, entered the fields of the messaging form with passed parameters, and then is ready to decide if the message should be sent, or further edited, or possibly cancelled. Appropriate MS storage is updated and subsequently processed as though the user manually started the create message interface and entered message information manually. Preferably, the compose command data is maintained to LBX History, a historical call log (e.g. incoming), or other useful storage for subsequent use. A standard email POP or mailbox interface, or a similar messaging interface, can be used. The message will arrive to a recipient like other messages. Various attributes can be set (e.g. special handling, NLS considerations, etc), but preferably, no confirmation of delivery status is requested in attributes since this is a broadcast.
213	O	See Send Command for identical processing.
215	C	Composing an application causes invocation of the application at the MS. The app parameter is preferably a fully qualified path name to the executable to start. In another embodiment, the app parameter is indirect: a path name to a "shortcut" (like a MS Windows shortcut). In another embodiment, the app parameter is an identifier string for the underlying operating system to know which application to start. The params parameter can be used for command line, or string to append, or pass, to the app/path parameter, for how to start the application (e.g. with parameters). Processing takes place as though the user manually launched the application (and with any optional params). Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
217	S	Composing a document causes invocation of the appropriate application at the MS in accordance with the object type as though the user selected the document for automatically being associated to the correct application when opening the document for composing (e.g. edit/manage) it. The doc parameter may be preferably a fully qualified path name to the document. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
219	S	Composing a file causes invocation of the appropriate application at the MS in accordance with the file type of the fully qualified path name of the file as though the user selected the file for automatically being associated to the correct application when opening the document. Processing takes place as though the user manually launched the application for the specified file to compose it. The path parameter is preferably a fully qualified path name to the file. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.

Fig. 65B-3

<i>Operand</i>	PM	<u>Preferred embodiment Compose processing</u>
221	O	Composing content causes invocation of the appropriate application at the MS in accordance with the content as though the user selected the content for automatically being associated to the correct application when opening the content. Processing takes place as though the user manually launched the applicable application for the content for composing (e.g. manage/edit of) it. The path parameter is preferably a fully qualified specification to the content. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
223	C	Composing a DB object causes invocation of the appropriate database query manager DB object creation interface to a context complementary to the type of DB object as though the user started the query manager and manually entered the DB object for creation. Processing takes place as though the user manually launched the query manager, entered the fields of the database object form, and then is ready to further work with the starting template of DB object information. In one embodiment, the DB-obj parameter contains directives for automatically populating specified data to a particular Query Manager create object interface. In another embodiment, the DB-obj parameter is specified for an existing DB object for then being opened by the query manager for further review or work. Appropriate MS storage is updated and subsequently processed as though the user had entered information manually. Preferably, the compose command data is maintained to LBX History, a historical call log (e.g. incoming), or other useful storage for subsequent use.
225	O	Composing data causes modifying the value of the data at the MS (analogous to a Notify data action -- set data to value). An error can result if the data is not resolvable for the attempt. In the preferred embodiment, the data is a global system variable visible to all processes of a MS operating system. In other embodiments, the data may have limited scope which is made accessible to present disclosure processing (e.g. with extern). A recognized AppTerm causes access to record 5300 for proper semaphore synchronized access. Preferably, the data affected is maintained to LBX History, a historical log (e.g. incoming), or other useful storage for subsequent user browse, or programmatic access, of the data variable name, its before and after values and date/time stamp of when sent, and for presentation of the data value in response to a user action to show it.
227	O	Composing a semaphore causes modifying the value of the semaphore at the MS (analogous to a Notify sem action -- set sem to value). An error can result if the semaphore is not resolvable for the attempt. In the preferred embodiment, the semaphore is a global system semaphore visible to all processes of a MS operating system. In other embodiments, the semaphore may have limited scope which is made accessible to present disclosure processing (i.e. RAM semaphore). Preferably, the semaphore value before and after setting is maintained to LBX History, a historical log (e.g. incoming), or other useful storage for subsequent user browse, or programmatic access, and for presentation of the semaphore information in response to a user action to show it.

Fig. 65B-4

<i>Operand</i> ↓	PM	<u>Preferred embodiment Compose processing</u>
229	S	Composing a directory causes invocation of the appropriate application (e.g. file system manager) at the MS as though the user selected the directory for automatically being associated to the correct file management application when opening the directory. Processing takes place as though the user manually launched the application for working with the directory. The path parameter is preferably a fully qualified path name to the directory. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
231	C	Composing an application context causes invocation of the application at the MS and then executing a macro within the application context (analogous to a Send app context action). The app parameter is preferably a fully qualified path name to the executable to start. In another embodiment, the app parameter is indirect: a path name to a "shortcut" (like a MS Windows shortcut). In another embodiment, the app parameter is an identifier string for the underlying operating system to know which application to start. The macro parameter is preferably a file, path, or accessible variable name containing a set of keystrokes that can be directed to standard/user-interface input. In another embodiment, the macro parameter is a prerecorded user input scenario (for play after application launched -- pulldown selections, mouse droppings, clicks, etc) captured to a file or stored in an accessible variable name. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
233	S	Composing a focused user interface object causes invocation of the appropriate application (e.g. graphic application by file type embodiment .jpg, .gif, etc) at the MS as though the user manually captured the focused user interface object (e.g. Alt-Prtsrn) using the first command string syntax parameter, invoked the correct graphical application to open for the captured image, and is ready for save of the file, or for further editing. Processing takes place as though the user manually launched the application for the specified file. The embodiment's file type preference may influence which application is to be launched. The first parameter can be used to change the keystroke sequence for capture. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
235	O	Composing user interface control causes redirecting the keystroke macro to user interface input of the MS as if it were entered by the MS user (analogous to a Send user interface control action). The macro parameter is preferably a file, path, or accessible variable name containing a set of keystrokes that can be directed to standard input. In another embodiment, the macro is a prerecorded user input scenario (for play after application launched -- pulldown selections, mouse droppings, clicks, etc) captured to a file or stored in an accessible variable name. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.

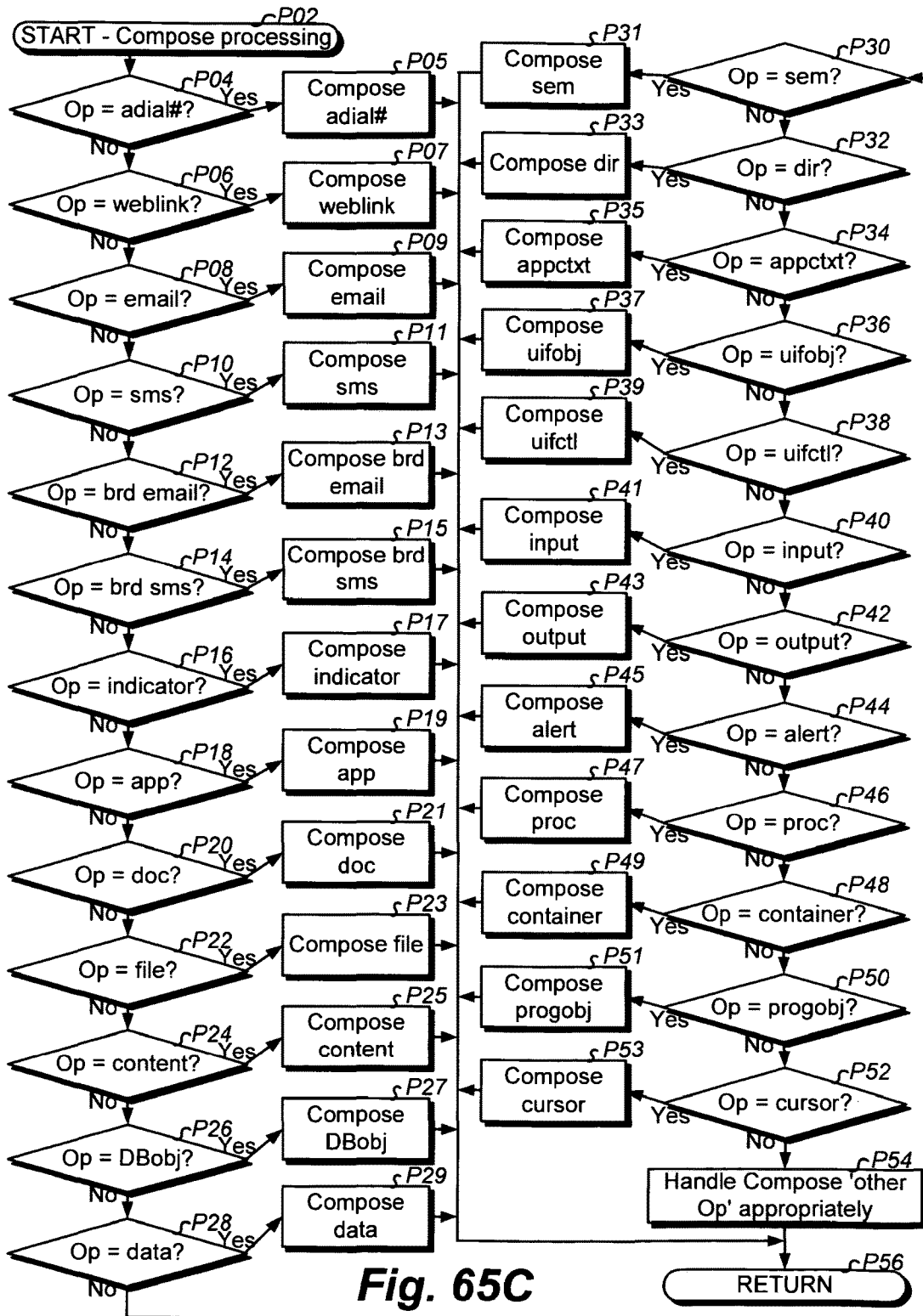
Fig. 65B-5

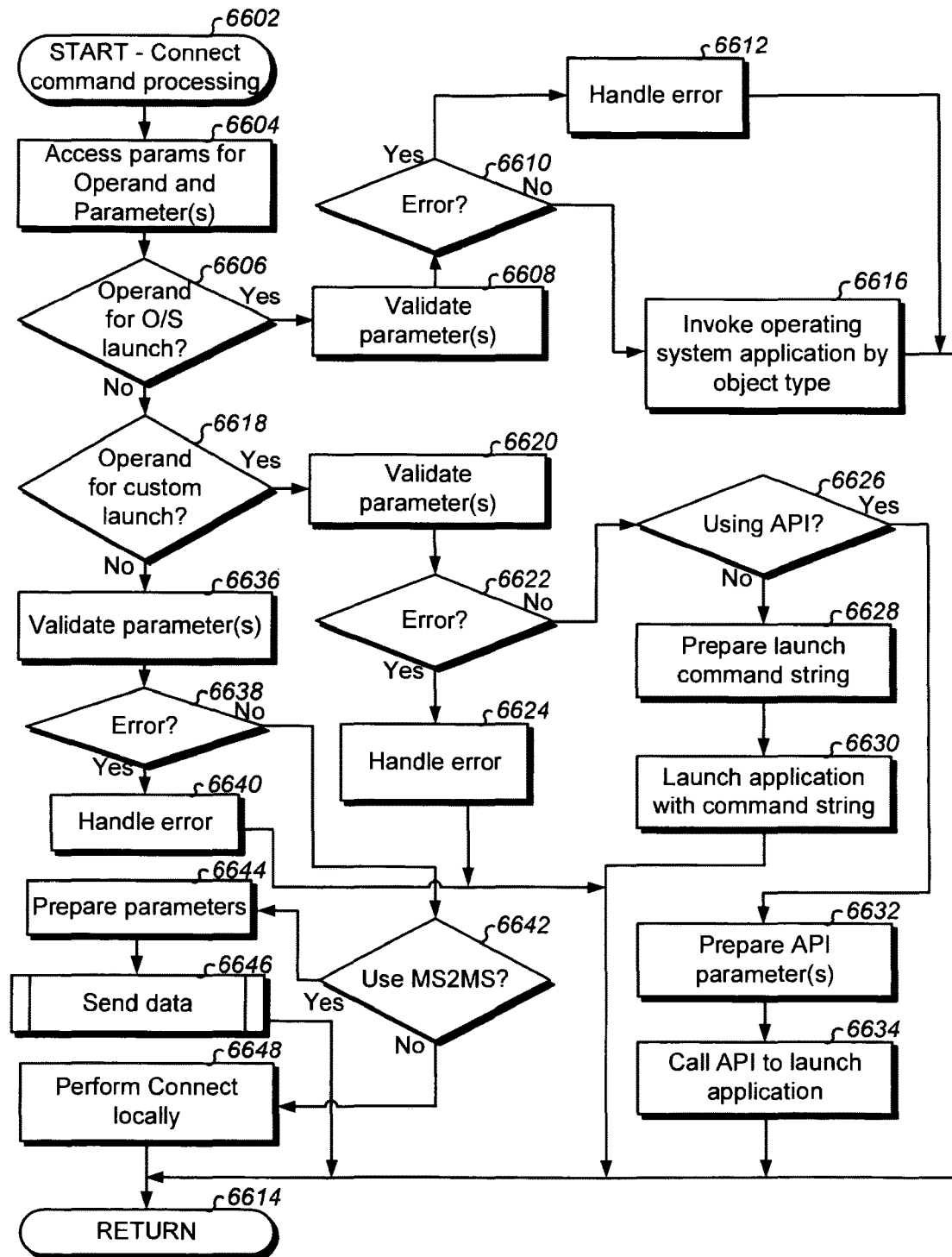
<i>Operand</i>	PM	<u>Preferred embodiment Compose processing</u>
237	O	Composing input causes redirecting the input to the iodev parameter input device stream of the MS as if it were entered by the MS user, or programmatically specified to the iodev I/O device parameter by a data processing system process (analogous to a Send input action). The input parameter is preferably a file or accessible variable name containing a datastream recognizable by the iodev connected device. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
239	O	Composing output causes redirecting the output to the iodev parameter output device stream of the MS as if it were entered by the MS user, or programmatically specified to the iodev I/O device parameter by a data processing system process (analogous to a Send output action). The output parameter is preferably a file or accessible variable name containing a datastream recognizable by the iodev connected device. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
241	S	Composing an alert causes invocation of the appropriate alerter application at the MS as though the user selected the alert application, manually entered the alert parameter, and is ready to decide what to do with the alert, for example send it with a minimal action (e.g. ok), edit it, or cancel it. Processing takes place as though the user manually launched the application for creating the specified alert. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
243	O	Composing a process causes sending an operating system signal (see UNIX signaling) to the process with Process ID (PID) of the pid parameter (analogous to a Notify process action). A numeric value parameter (e.g. 0 or 1) may be communicated with the signal. An error can be logged if the process is not found for signaling. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
245	S	Composing a container causes launch of a MS container manager (e.g. file manager) to examine the contents of the container having the path in the container parameter (e.g. c:\dir1\subdir3) (analogous to a Notify container action). An error is logged if the file manager is not found for launch, or if the container is invalid. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
247	O	Composing a program object causes launch of a MS development environment application (e.g. Microsoft Visual Studio or IBM C-Set development consoles, etc), performing a search for the progobj parameter symbol, and producing search results of all occurrences for the current development working directory, mount point, or last used development repository. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.

Fig. 65B-6

<i>Operand</i>	PM	Preferred embodiment Compose processing
249	O	Composing a cursor causes invocation of the appropriate application (e.g. graphic application by file type embodiment .bmp, .ico, etc) at the MS as though the user manually launched the application for the cursor parameter, and is ready for save of the file, or for further editing, or for cancellation. Depending on the cursor parameter referenced, an appropriate application will be launched for graphics, animation, etc. The cursor parameter is preferably a fully qualified path to determine the cursor (e.g. file). Processing takes place as though the user manually launched the application for the specified file. The embodiment's file type preference will influence which application is to be launched. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
251	S	Composing a calendar object causes interface to the calendar system for invoking the create calendar object interface and defaulting the appropriate calendar interface fields with the passed parameters. The calendar object parameter may be as described for Send calendar object, except for defaulting calendar interface create object interface(s). The user can subsequently create the scheduled event with little effort, or after optional modification, with a minimal action (e.g. click ok). Processing takes place as though the user manually launched the calendar application, entered the fields of the calendar form with passed parameters, and then is ready to decide what to do with it. Appropriate MS storage is updated and subsequently processed as though the user had manually started the calendar application and entered the calendar information manually. Preferably, the compose command data is maintained to LBX History, a historical call log (e.g. incoming), or other useful storage for subsequent use. A standard calendaring interface is preferably used. Attributes can be set as is customary for a calendar object.
253	S	Composing an address book (AB) object causes interface to the AB system for invoking the create AB object interface and defaulting the appropriate AB interface fields with the passed parameters. The AB object parameter may be as described for Send AB object, except for defaulting AB interface create object interface(s). The user can subsequently create the AB entry with little effort, or after optional modification, with a minimal action (e.g. click ok). Processing takes place as though the user manually launched the AB application, entered the fields of the AB form with passed parameters, and then is ready to decide what to do with it. Appropriate MS storage is updated and subsequently processed as though the user manually started the AB application and entered the AB information manually. Preferably, the compose command data is maintained to LBX History, a historical call log (e.g. incoming), or other useful storage for subsequent use. A standard AB interface is preferably used. Attributes can be set as may be customary for an AB entry.
...		

Fig. 65B-7



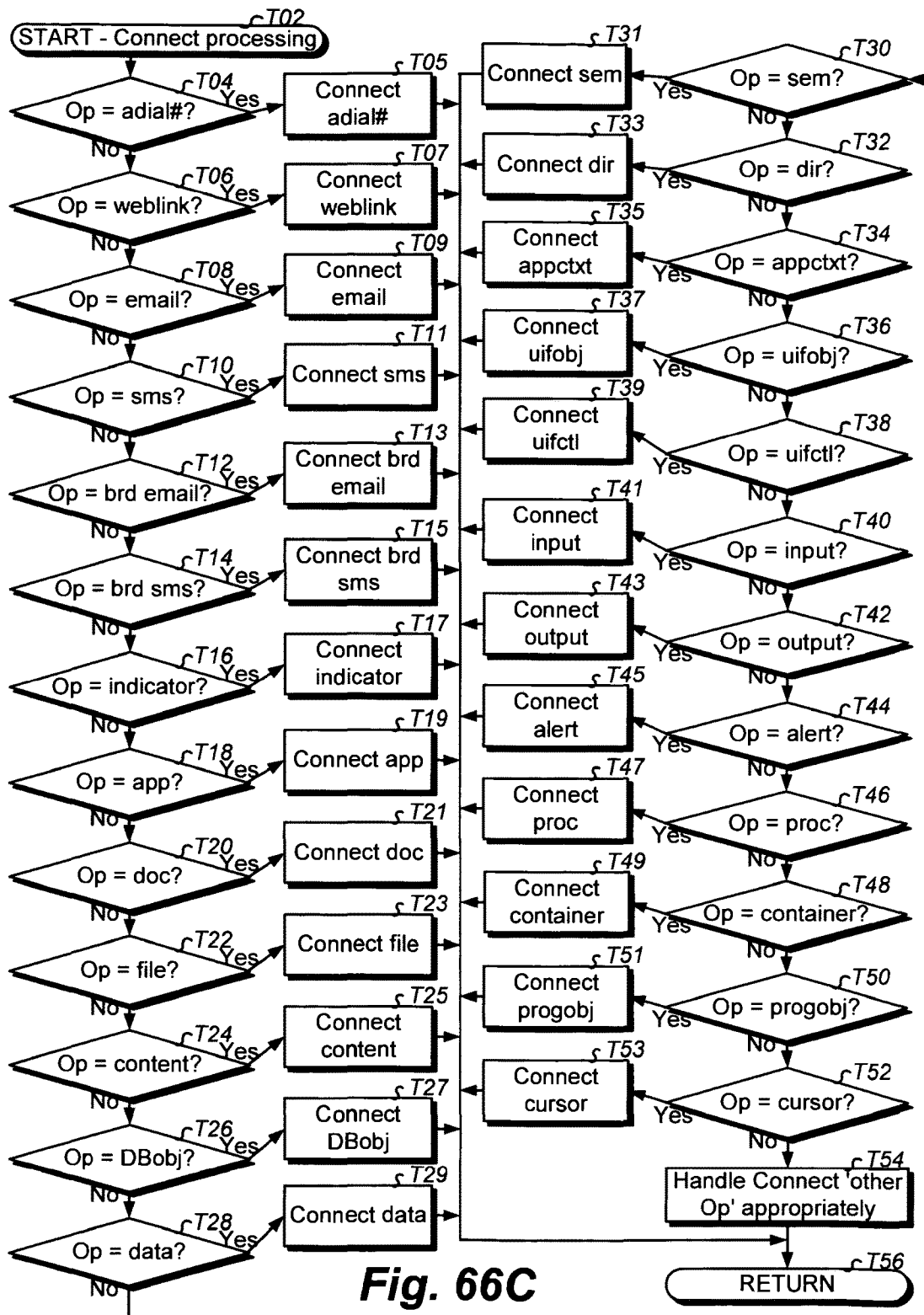
**Fig. 66A**

<i>Operand</i>	PM	Preferred embodiment Connect processing
201	C	Connecting with an auto-dial # launches the MS phone number calling interface with the auto-dial # parameter defaulted for placing a call (like Notify autodial #). The call is actually made as though the user manually launched the dialing application, entered the auto-dial # and then chose to make the call with it. Appropriate MS storage is updated and subsequently processed as though the user had entered the # for calling manually and then made the call. Conventional call processing takes place thereafter. Preferably, the connect command data is maintained to LBX History, a historical call log (e.g. outgoing), or other useful storage for subsequent use.
203	S	Connecting with a web link launches a MS browser and invokes (transposes to) the link as though the user had entered it manually and went to the weblink page (like Notify weblink). In one embodiment, the weblink parameter includes URL parameter(s). In another embodiment, the params parameter supports a URL command string for appending to the weblink (e.g. "?v=yes&T=go") for customized web page processing. An alternate embodiment can fire form variables for active web page processing using the params parameter. Processing takes place as though the user manually launched the browser application, entered the weblink and then loaded the weblink webpage. Appropriate MS storage is updated and subsequently processed as though the user had entered the weblink in the browser manually. Preferably, the connect command data is maintained to LBX History, a historical log (web page load history), or other useful storage for subsequent use.
205	C	See Send Command for identical processing.
207	C	See Send Command for identical processing.
209	C	See Send Command for identical processing.
211	C	See Send Command for identical processing.
213	O	See Send Command for identical processing.
215	C	See Compose command for identical processing.
217	S	See Compose command for identical processing.
219	S	See Compose command for identical processing.
221	O	See Compose command for identical processing.
223	C	See Notify command for identical processing, except some applicable parameters not used.
225	O	See Compose command for identical processing.
227	O	See Compose command for identical processing.
229	O	See Compose command for identical processing.
231	C	See Compose command for identical processing.
233	S	See Compose command for identical processing.
235	O	See Compose command for identical processing.

Fig. 66B-1

<i>Operand</i> ↓	PM	<u>Preferred embodiment Connect processing</u>
237	O	See Compose command for identical processing.
239	O	See Compose command for identical processing.
241	C	Connecting with an alert causes interfacing to the alert subsystem for instantly producing the alert at the MS. Preferably, the connect command data is maintained to LBX History, a log, or other useful storage for subsequent use. The alert parameter of Notify processing is identical.
243	O	See Compose command for identical processing.
245	S	See Compose command for identical processing.
247	O	See Notify command for identical processing.
249	O	See Notify command for identical processing.
251	C	See Send Command for identical processing.
253	C	See Send Command for identical processing.
...		

Fig. 66B-2



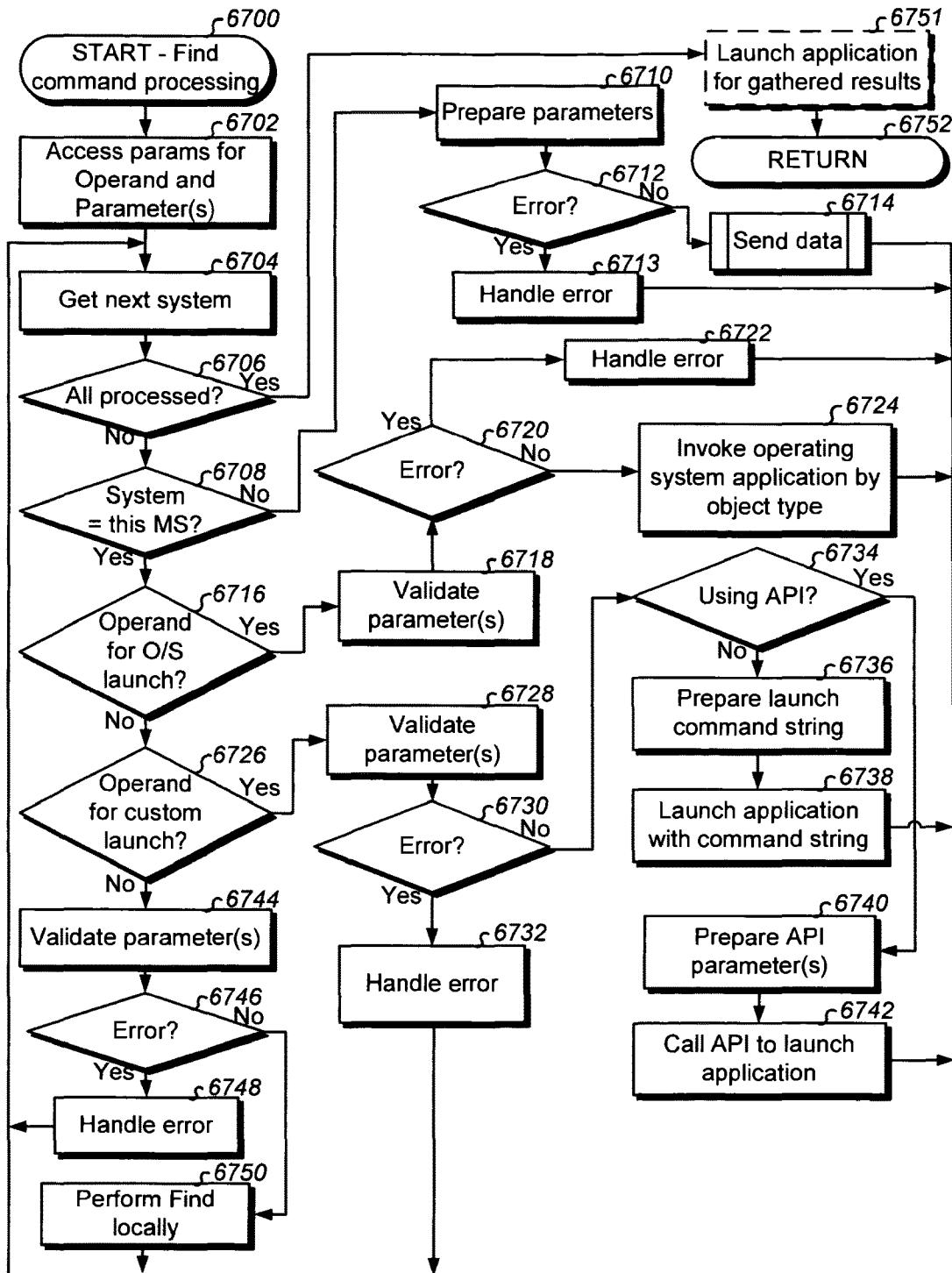


Fig. 67A

<i>Operand</i> ↓	PM	<u>Preferred embodiment Find processing</u>
201	C	Finding an auto-dial # launches a system (e.g. MS) phone number log interface with the auto-dial # parameter for searching. Preferably, both the outgoing and incoming logs are searched. The auto-dial # parameter can be a wildcard (pattern) for matching. In one embodiment, all matching occurrences found in history are presented with their date/time stamps, and perhaps other information, of the call and when it took place. In another embodiment, the most recent occurrence from a particular log is presented, and perhaps in an interface which enables calling the # with a minimal user action. The search takes place as though the user manually launched the search, entered the auto-dial # for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user manually performed the search. Preferably, the find cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. A new parameter can be specified for which log to search.
203	S	Finding a weblink launches a search to system (e.g. MS) browser history with the weblink parameter (and with the params parameter if specified) for searching. The weblink parameter can be a wildcard (pattern), and may include URL parameters, for matching. In one embodiment, all matching occurrences found in history are presented with their date/time stamps, and perhaps other information, of the link and when it was invoked. In another embodiment, the most recent occurrence from a particular invocation is presented, and perhaps in an interface which enables invoking (transposing to) the weblink with a minimal user action. In a preferred embodiment, the params parameter tells find processing how and where to search. The search takes place as though the user manually launched the search, entered the weblink for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. A new parameter can be specified for which folder to search.

Fig. 67B-1

Operand ↓	PM	<u>Preferred embodiment Find processing</u>
205	C	<p>Finding an email causes searching a system (e.g. MS) email system with search criteria of the email parameter string. The email parameter string can specify searching any email fields for any values including wildcarding (patterns for matching). Each field is referenced with a predefined name and then associated with a search criteria. For example, the email string of "subj:'personnel'; recip:'george@alltell.com'; body:'reduction in force'" causes searching all emails with a subject containing "personnel" and was sent to "george@alltell.com" and has a message body containing the string "reduction in force". To search for certain email containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:sent,inbox,company;" indicates to only search the email folders of sent, inbox, and company (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of email. Wildcarding (pattern matching) is preferably inherent by searching for substrings. All occurrences found in history are presented with at least their date/time stamps, subject line, sender and recipient, and perhaps other information, of the email and when it took place. In another embodiment, the most recent occurrence from searched folders is presented, and perhaps in an interface which enables appropriate MS email system processing from that point forward (e.g. when processed at local MS). Alternatively, an additional parameter indicates how to search. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.</p>

Fig. 67B-2

<i>Operand</i> ↓	<u>PM</u>	<u>Preferred embodiment Find processing</u>
207	C	<p>Finding an sms message causes searching a system (e.g. MS) sms messaging system with search criteria of the sms message parameter string. The message parameter string can specify searching any message fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria. For example, the sms message string of "recip:'2144034071@nextel.com,9725397137@lbxsrv.com';" causes searching all messages to the sought recipients. To search for certain messaging containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:outgoing" indicates to only search the outgoing folder (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of messages. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In the preferred embodiment, all occurrences found in history are presented with at least their date/time stamps, message, sender and recipient, and perhaps other information, of the message and when it took place. In another embodiment, the most recent occurrence from searched folders is presented, and perhaps in an interface which enables appropriate MS messaging system processing from that point forward (e.g. when processed at local MS). Alternatively, an additional parameter indicates how to search. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.</p>

Fig. 67B-3

<i>Operand</i> ↓	<u>PM</u>	<u>Preferred embodiment Find processing</u>
209	C	<p>Finding a broadcast email causes searching a system (e.g. MS) email system with search criteria of the email param string. The email param string can specify searching any email fields for any values including wildcarding. Each field is referenced with a predefined name and associated with a search criteria. For example, the param of "subj:'personnel'; recip:'george@alltell.com'; body:'reduction in force'" searches emails with a subject containing "personnel" and sent to "george@alltell.com" and has an email body containing "reduction in force". To search for certain email containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:sent, inbox, company;" indicates to only search the email folders of sent, inbox, and company (no specification preferably indicates to search all folders)). Those skilled in the art recognize useful syntaxes for searching any characteristics of email. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In the preferred embodiment, all occurrences found in history are presented with at least their date/time stamps, subject line, sender and recipient, and perhaps other info, of the email and when it took place. In another embodiment, the most recent occurrence from searched folders is presented, and perhaps in an interface which enables appropriate MS email system processing from that point forward (e.g. when processed at local MS). Alternatively, an additional parameter indicates how to search. The search takes place as though the user manually launched the search, entered criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.</p>

Fig. 67B-4

<i>Operand</i> ↓	PM	<u>Preferred embodiment Find processing</u>
211	C	Finding a broadcast sms message causes searching a system (e.g. MS) sms messaging system with search criteria of the sms message param string. The message param string can specify searching any message fields for any values including wildcarding. Each field is referenced with a predefined name and associated with a search criteria. For example, the sms message string of "recip:'2144034071@nextel.com,9725397137@lboxsrv.com';" causes searching messages to the sought recipients. To search for certain messaging containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:outgoing" indicates to only search the outgoing folder (no specification preferably indicates to search all folders)). Those skilled in the art recognize useful syntaxes for searching any characteristics of messages. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In the preferred embodiment, all occurrences found in history are presented with at least their date/time stamps, message, sender and recipient, and perhaps other information, of the message and when it took place. In another embodiment, the most recent occurrence from searched folders is presented, and perhaps in an interface which enables appropriate MS messaging system processing from that point forward (e.g. when processed at local MS). Alternatively, an additional parameter indicates how to search. The search takes place as though the user manually launched the search, entered search criteria, and then was presented with result(s). Appropriate MS storage is updated and processed as though the user manually performed the search. Preferably, find cmd data is maintained to LBX History, a historical log, or other useful storage for later use.
213	O	Finding an indicator searches appropriate system (e.g. MS) storage for the indicator (e.g. storage/memory used for indicators by other commands). The indicator parameter string specifies the indicator (e.g. string) being sought and wildcarding is supported. Any active user interface object containing the indicator is surfaced. If more than one user interface object contains the indicator, then all objects are appropriately tiled with the most recent in the priority position(s). In another embodiment, appropriate MS storage/memory which contains the history of indicators sent is searched and all occurrences found in history are presented with at least their date/time stamps, the indicator, and perhaps other information. In yet another embodiment, a new parameter tells processing whether to surface/prioritize active objects, or to search history for when indicators were sent. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.

Fig. 67B-5

<i>Operand</i> ↓	PM	Preferred embodiment Find processing
215	C	Finding an application causes searching the system (e.g. MS) for the application (and with the params parameter if specified). The app parameter is preferably an executable name, optionally with parameters that were passed. Providing a partial or full path to the application parameter will validate that it is found there. The app parameter string preferably supports wildcarding. Embodiments (or as specified with params and/or new parameters) include file system searching, invocation history (e.g. Microsoft Windows up/down arrow command line recall) searching for what had been invoked (perhaps within a trailing time period), what is currently running, what has been terminated (perhaps within a trailing time period), or any of these for a particular invoked identity, credentials, or owner. In the preferred embodiment, all occurrences found on the MS and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information. In another embodiment, all parts which are linked to the executable are identified with their paths, date/time stamps, size, and perhaps attributes when a symbol file is specified with a new parameter. The symbol file is output from a link process and can be used to identify all executable parts such as dynamic link libraries, linked binaries, and any other executable binary file involved with the application. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and processed as though the user had manually performed the search. Preferably, find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
217	S	Finding a document causes searching the system (e.g. MS) for the document. The doc parameter is a document name. The document parameter can be a wildcard (pattern) for matching. Providing a partial or full path to the document name will validate that it is found there. In the preferred embodiment, all occurrences found on the MS and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
219	S	Finding a file causes searching the system (e.g. MS) for the file. The path parameter is a file name. Providing a partial or full path to the file will validate that it is found there. The path parameter can be a wildcard (pattern) for matching. In the preferred embodiment, all occurrences found on the MS and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for

Fig. 67B-6

Operand ↓	PM	Preferred embodiment Find processing
		subsequent use.
221	O	Finding content causes searching the system (e.g. MS) for the content. The content parameter can be a wildcard (pattern) for matching. The content parameter can be a handle to the content, or a search criteria for the content. In the preferred embodiment, all occurrences found on the MS and where the content is located is presented to the user, perhaps with other content description information. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). There are various embodiments for how and where content is maintained. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
223	C	Finding a DB object causes searching the system (e.g. MS) database(s) for the database object. The database object parameter is provided with a variety of syntaxes depending on the type of database object sought. For example, the DB-obj parameters is "T:tablename" to seek a table, "S:schemaname" to seek a particular schema, "C:columnname" to seek a particular column name, "D:DBname" to seek a particular DB name, "R:rolename" to seek a particular role set, "P:procname" to search for particular stored procedure, etc. There are unique syntaxes for every type of DB object being sought. Those skilled in the art know how to query system tables for particular DB object(s) sought. An appropriate SQL client API should be used. If necessary, an additional parameter is specified for authentication credentials (may be specified with DB-obj string syntax). The search criteria can be a wildcard (pattern) for matching. In the preferred embodiment, all occurrences found on the MS and information about the occurrence is presented to the user. The search takes place as though the user manually launched the search, entered the criteria or query for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.

Fig. 67B-7

Operand ↓	PM	Preferred embodiment Find processing
225	O	Finding data causes searching the system (e.g. MS) for the data. In the preferred embodiment, the data is a global system variable visible to processes of a MS O/S. In other embodiments, the data may have limited scope which is made accessible to present disclosure processing (e.g. with extern). Depending on the embodiment, data may be that which is contained in a program data segment, stack segment, and/or extra segment. There can be unique syntaxes for specifying which type of data is being sought (e.g. "S:dataname"). The search criteria can be a wildcard (pattern) for matching. In the preferred embodiment, all occurrences found on the MS and information about the occurrence including its current value is presented to the user. In one embodiment, a well known location of link symbol information files are consulted, and in another embodiment a new parameter specifies where to look, or which symbol file of information to use. An AppTerm uses record 5300 for access. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, find command data is maintained to LBX History, a historical log, or other useful storage for later use.
227	O	Finding a semaphore causes reading the current value of the semaphore at the system (e.g. MS) where the find command action is being executed and then presenting the current value along with any other useful information for the semaphore. The semaphore param can be a wildcard (pattern) for matching. In the preferred embodiment, the semaphore is a global system semaphore visible to all processes of a MS operating system. In other embodiments, the semaphore may have limited scope which is made accessible to present disclosure processing. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
229	S	Finding a directory causes searching the system (e.g. MS) for the directory (path). The path parameter is a directory name. Providing a more defined partial or full path to the path parameter will narrow down the results if the directory exists in more than one place. The path parameter can be a wildcard (pattern) for matching. In the preferred embodiment, all occurrences found on the MS and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.

Fig. 67B-8

<i>Operand</i> ↓	PM	<u>Preferred embodiment Find processing</u>
231	C	Finding an application context causes invocation of the application at the system (e.g. MS) and then executing a macro within the application context (similar to Compose app object processing). The app parameter is preferably a fully qualified path name to the executable to start. In another embodiment, the app parameter is indirect: a path name to a "shortcut" (like a MS Windows shortcut). In another embodiment, the app parameter is an identifier string for the underlying operating system to know which application to start. The search criteria can be a wildcard (pattern) for matching. The macro parameter is preferably a file, or path, or accessible variable name containing a set of keystrokes that can be directed to standard/user-interface input. In another embodiment, the macro parameter is a prerecorded user input scenario (for play after application launched -- pulldown selections, mouse droppings, clicks, etc) captured to a file or stored in an accessible variable name. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
233	S	Finding a user interface object causes finding and focusing the user interface object at the system (e.g. MS) which contains the object text (objtxt) parameter. In a preferred embodiment, there is a unique syntax for which places of user interface objects that are currently active are to be search (e.g. title bar, entry fields, radio button options, window text, combinations thereof, etc). The search criteria can be a wildcard (pattern) for matching. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
235	O	Finding user interface control causes searching the system (e.g. MS) storage and/or memory which was used for processing another command (e.g. Compose) to redirect the keystroke macro to standard input of the MS as if it were entered by the MS user. The search criteria can be a wildcard (pattern) for matching. The macro parameter is the same as was used by the command and is to be matched. In the preferred embodiment, presented in the search results are all occurrences of previous command actions, which used the macro at the MS, including the command, date/time stamp, and other information recorded (e.g. to LBX History, a historical log, or other useful storage for subsequent use). The search takes place as though the user manually launched a search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, historical log, or other useful storage for subsequent use.

Fig. 67B-9

<i>Operand</i> ↓	PM	Preferred embodiment Find processing
237	O	Finding input causes searching the system (e.g. MS) storage and/or memory which was used for processing another command (e.g. Compose) to redirect the input to the iodev device of the MS. The iodev and input parameters are the same as was used by a previous command and is to be matched. The search criteria can be a wildcard (pattern) for matching. In the preferred embodiment, presented in the search results are all occurrences of previous command actions, which used the iodev and input at the MS, including the command, date/time stamp, and other information recorded (e.g. to LBX History, a historical log, or other useful storage for subsequent use). The search takes place as though the user manually launched a search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
239	O	Finding output causes searching the system (e.g. MS) storage and/or memory which was used for processing another command (e.g. Compose) to redirect the output to the iodev device of the MS. The search criteria can be a wildcard (pattern) for matching. The iodev and output parameters are the same as was used by a previous command and is to be matched. In the preferred embodiment, presented in the search results are all occurrences of previous command actions, which used the iodev and output at the MS, including the command, date/time stamp, and other information recorded (e.g. to LBX History, a historical log, or other useful storage for subsequent use). The search takes place as though the user manually launched a search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
241	S	Finding an alert causes searching the system (e.g. MS) for the alert. The alert parameter is the same parameter used to generate an alert (e.g. using another command). In the preferred embodiment, all occurrences found on the MS which is associated to the alerter application in use at the MS, and which is used for other commands disclosed, are presented to the user with at least their date/time stamps, and perhaps other information. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). The search criteria can be a wildcard (pattern) for matching. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.

Fig. 67B-10

<i>Operand</i>	PM	<u>Preferred embodiment Find processing</u>
243	O	Finding a process causes finding all process names running at the system (e.g. MS) which contain the pname string parameter (e.g. in UNIX: "ps -ef grep pname"). In the preferred embodiment, all occurrences found running at the MS are presented with interesting programmatic information such as when started, its size, etc (see UNIX ps command for other information that can be presented here in various embodiments; an additional parameter (like ps parameters) can specify what info to provide to the user). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, historical log, or other useful storage for subsequent use.
245	S	Finding a container causes searching the system (e.g. MS) for the container. The container parameter is a container name (e.g. file system directory) depending on the MS or environment. Unique syntaxes can be used for which type of container is being searched. In the preferred embodiment, all occurrences found on the MS and their paths are presented to the user with information of interest. The search criteria can be a wildcard (pattern) for matching. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, historical log, or other useful storage for subsequent use.
247	O	Finding a program object causes searching the system (e.g. MS) for the program object. In the preferred embodiment, a unique syntax is used for which type of program object is being sought (e.g. Q:queue name has a queue qualifier prefix). There can be unique syntaxes for specifying which type of program object is being sought (e.g. "S:dataname"). In the preferred embodiment, all occurrences found on the MS and information about the occurrence including its current value is presented to the user. A null data parameter returns all occurrences found. A non-null data parameter returns these objects having the data value. Objects must be programmatically accessible. In one embodiment, a well known location of link symbol information files are consulted, and in another embodiment a new parameter specifies where to look, or which symbol file of information to use. In another embodiment, MS storage and/or memory is searched which recorded a previous atomic command action, and the search takes place for a previous command(s) (e.g. Notify) for when performed and what was performed. The search criteria can be a wildcard (pattern) for matching. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.

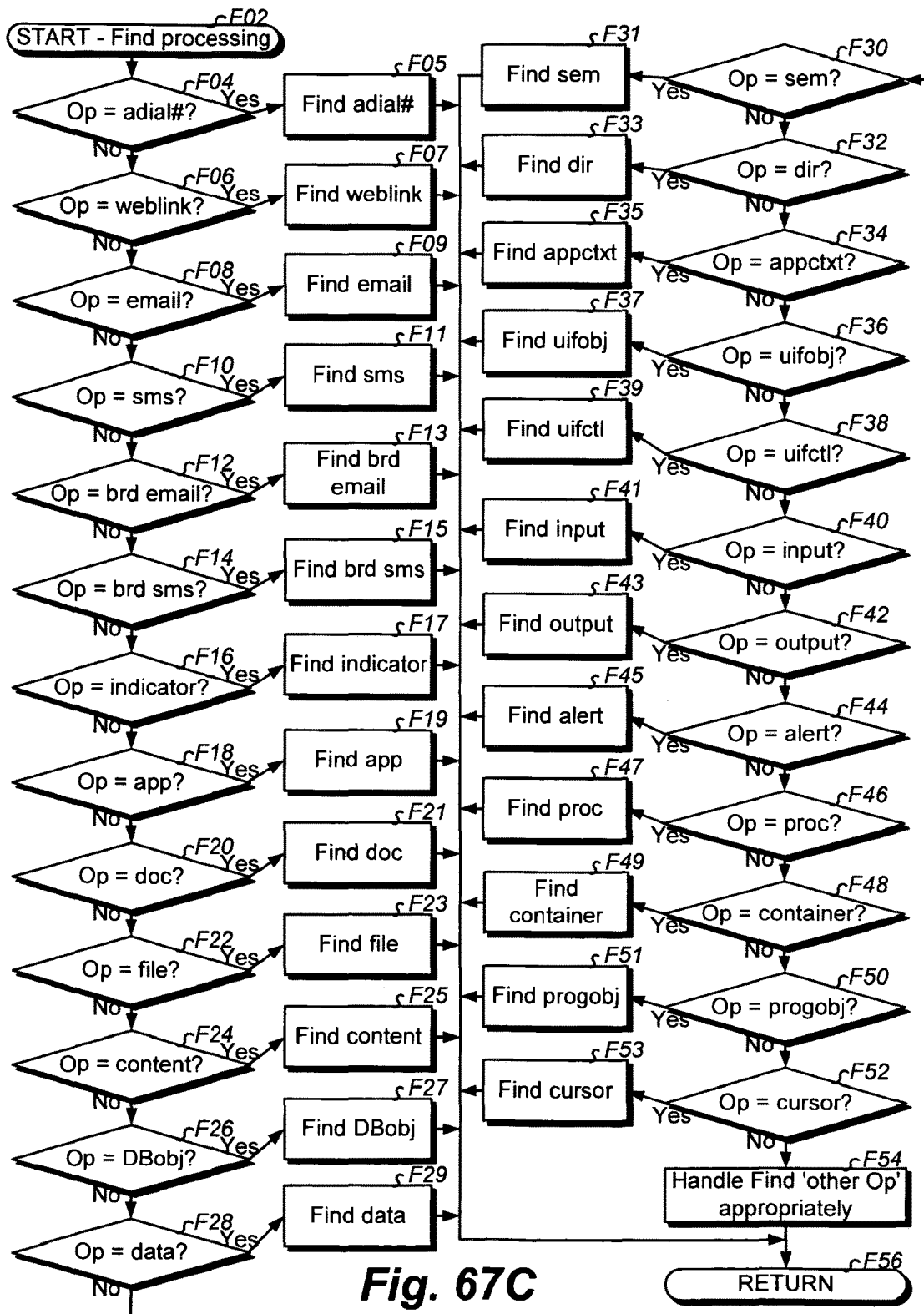
Fig. 67B-11

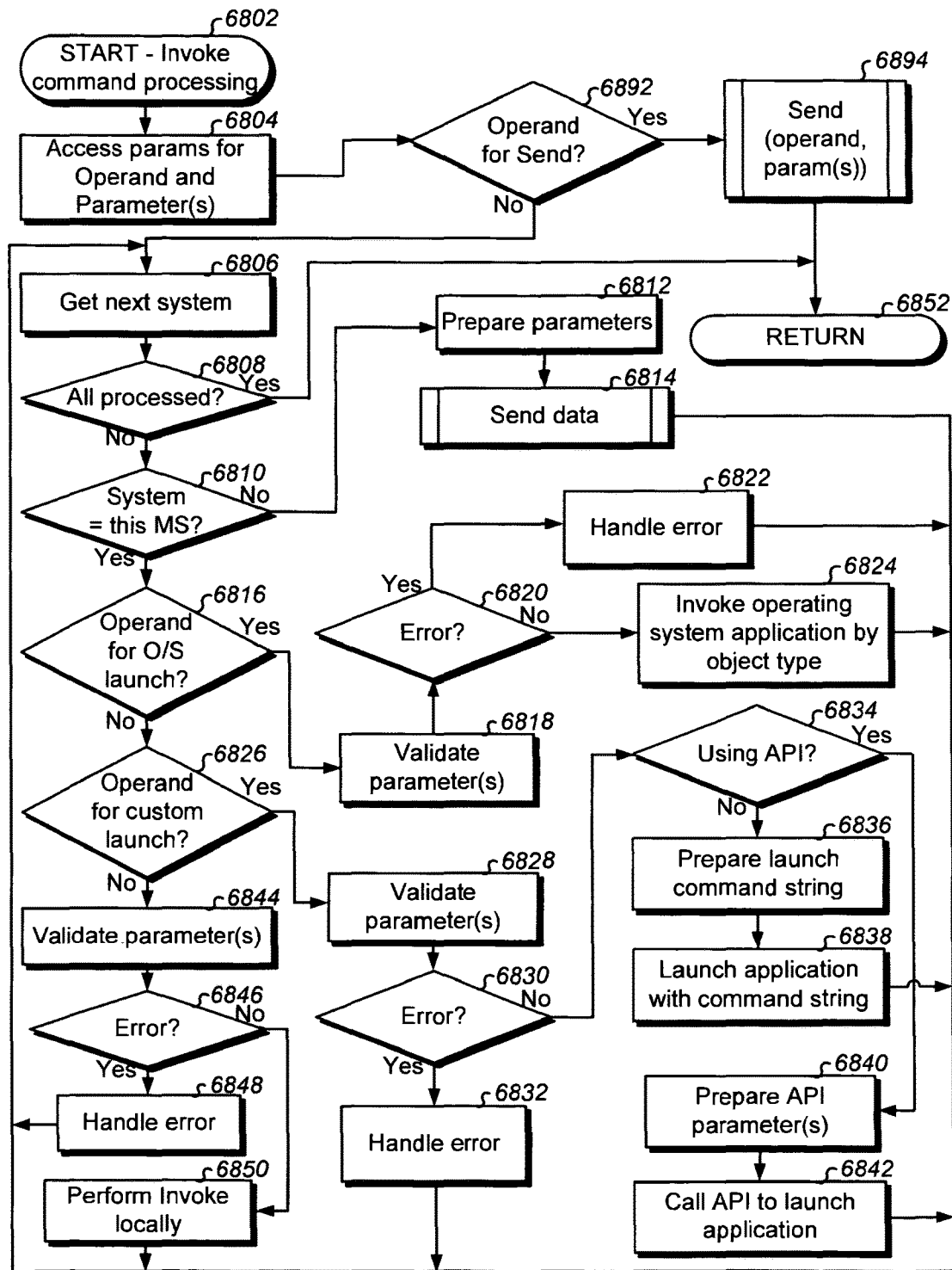
<i>Operand</i> ↓	<u>PM</u>	<u>Preferred embodiment Find processing</u>
249	O	Finding a cursor causes searching the system (e.g. MS) storage and/or memory which was used for processing another command (e.g. Compose) to view or alter a cursor. In the preferred embodiment, presented in the search results are all occurrences of previous command actions, which viewed or altered the cursor at the MS, including the command, date/time stamp, and other information recorded (e.g. to LBX History, a historical log, or other useful storage for subsequent use). The search criteria can be a wildcard (pattern) for matching. The search takes place as though the user manually launched a search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
251	C	Finding a calendar object causes searching a system (e.g. MS) calendar system with search criteria of the calendar object parameter string. The calendar object parameter string can specify searching any calendar entry fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria (similar to email above). Those skilled in the art recognize useful syntaxes for searching any characteristics of calendar objects with an appropriate syntax. The calendar object parameter is at least a string with a syntax for querying any combination of calendar object fields. In the preferred embodiment, all occurrences found in history are presented with at least their date/time stamps, attendees, and perhaps other information, of the calendar object and when it was scheduled. In another embodiment, the most recent occurrence from the calendaring system is presented, and perhaps in an interface which enables appropriate MS calendar system processing from that point forward (e.g. when processed at local MS), or alternatively an additional parameter can specify how to search. Wildcarding (pattern matching) is preferably inherent by searching for substrings. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.

Fig. 67B-12

<i>Operand</i> ↓	<u>PM</u>	<u>Preferred embodiment Find processing</u>
253	C	Finding an address book (AB) object causes searching a system (e.g. MS) AB system with search criteria of the AB object parameter string. The AB object parameter string can specify searching any AB entry fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria (similar to email above). Those skilled in the art recognize many useful syntaxes for searching any characteristics of AB objects/entries with an appropriate syntax. The AB object parameter is at least a string with a syntax for querying any combination of AB object fields. In the preferred embodiment, all occurrences found are presented with appropriate AB information. In another embodiment, the most recent occurrence from the AB system is presented, and perhaps in an interface which enables appropriate MS AB system processing from that point forward (e.g. when processed at local MS), or alternatively an additional parameter can specify how to search. Wildcarding (pattern matching) is preferably inherent by searching for substrings. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
...		

Fig. 67B-13



**Fig. 68A**

<i>Operand</i> ↓	PM	<u>Preferred embodiment Invoke processing</u>
201	C	Invoking with an auto-dial # launches the MS phone number calling interface at the MS (local or remote) with the auto-dial # parameter for placing a call (like/see Notify / Connect autodial # processing). The call is actually made as though a user manually launched the dialing application at the particular MS, entered the auto-dial # and then chose to make the call with it. Appropriate MS storage is updated and subsequently processed as though the user had entered the # for calling manually, and then make the call. Conventional call processing takes place thereafter. Preferably, the invoke command data is maintained to LBX History, a historical call log (e.g. outgoing), or other useful storage for subsequent use. A system parameter provides means for placing the call from another system (e.g. another MS) – like a Host specification.
203	S	Invoking with a web link launches the MS browser at the particular MS and invokes (transposes to) the link as though the user had entered it manually and went to the weblink page (like/see Notify / Connect weblink processing). In one embodiment, the weblink parameter includes URL parameter(s). In another embodiment, the params parameter supports a URL command string for appending to the weblink (e.g. "?v=yes&T=go") for customized web page processing. An alternate embodiment can fire form variables for active web page processing using the params parameter, or URL parameter(s). Processing takes place as though the user manually launched the browser application at the particular MS, entered the weblink and then loaded the weblink webpage. Appropriate MS storage is updated and subsequently processed as though the user had entered and invoked the weblink in the browser manually. Preferably, the compose command data is maintained to LBX History, a historical log (web page load history), or other useful storage for subsequent use.
205	E	See Send Command for identical processing.
207	E	See Send Command for identical processing.
209	E	See Send Command for identical processing.
211	E	See Send Command for identical processing.

Fig. 68B-1

<u>Operand</u>	<u>PM</u>	<u>Preferred embodiment Invoke processing</u>
213	O	Invoking an indicator updates the appropriate MS storage so that the currently focused user interface object (e.g. window titlebar) of the particular MS user interface is modified with the indicator (like/see Send indicator processing). If there are no active user interface objects in the MS user interface, then an appropriate alert area of the currently focused interface is to display the indicator. The user can clear (remove) the indicator when desired. Preferably, the indicator is used for modifying other focused objects (e.g. titlebars) or other focused areas in the user interface so as to not get overlooked. For example, as the user navigates and surfaces/focuses new user interface objects, the indicator remains visible on the newly focused object. Preferably, the indicator is selectable by the user of the MS for showing all other send command parameters associated, as well as a date/time stamp of when sent. In other embodiments, the most recently displayed indicator is displayed in the appropriate focused area, but the user can conveniently select any indicators which were sent in history at some point in time for sought indicator information by selecting the currently displayed indicator and then requesting to browse/scroll history of previously delivered indicators (with options to see details). Preferably, the invoke command data is maintained to LBX History, a historical log (web page load history), or other useful storage for subsequent use.
215	C	Invoking an application causes invocation of the application at the particular MS (like/see Send app processing). The app parameter is preferably a fully qualified path name to the executable to start, and may already include parameters. In another embodiment, the app parameter is indirect: a path name to a "shortcut" (like a MS Windows shortcut). In another embodiment, the app parameter is an identifier string for the underlying operating system to know which application to start. The params parameter may specify the executable parameters, or may be used for how to start the application (like attributes of Send app processing). An error is logged if the app parameter is not found for launch. Preferably, the invoke command data is maintained to LBX History, a historical log (web page load history), or other useful storage for subsequent use.
217	S	Invoking a document causes invocation of an appropriate application at the particular MS in accordance with the object type as though the user selected the document for automatically being associated to the correct application when opening the document. The user can then decide what to do with the document once it is opened in the appropriate application. In an alternate embodiment, an additional parameter is provided for exactly what to do with the document, in which case an appropriate API is invoked with the document (i.e. PM = C). The doc parameter is preferably a fully qualified path name to the document. Preferably, the invoke command data is maintained to LBX History, a log, or other useful storage for subsequent use.

Fig. 68B-2

<i>Operand</i> ↓	PM	<u>Preferred embodiment Invoke processing</u>
219	S	Invoking a file causes invocation of an appropriate application at the particular MS in accordance with the file type as though the user selected the file for automatically being associated to the correct application when opening the document. Processing takes place as though the user manually launched the application for the specified file. The user can then decide what to do with the file once it is opened in the appropriate application. In an alternate embodiment, an additional parameter is provided for exactly what to do with the file, in which case an appropriate API is invoked with the file (i.e. PM = C). The path parameter is preferably a fully qualified path name to the file. Preferably, the invoke command data is maintained to LBX History, a log, or other useful storage for subsequent use.
221	O	Invoking content causes the content to be presented at the particular MS in a manner which is appropriate for the content type (like/see Notify content processing). The content parameter is one that cannot be classified in the other operands, but is content for presentation nevertheless. Examples include special data records (e.g. extern variable name), content data memory locations (e.g. programmatic variable), or files containing a customizably processed format. Methods of displaying the content include audio and/or visual using applicable MS capabilities. Preferably, the invoke command data is maintained to LBX History, a historical content log (e.g. incoming), browser history data, or other useful storage for subsequent user browse of the accompanying content and a date/time stamp of when sent, and for presentation of the content. Various embodiments will save to LBX History how many times, and when, the content was presented.
223	O	Invoking a Database (DB) object causes the DB object (i.e. qualified database with access query string) to be modified with the query parameter at the particular MS (like/see Notify DB-obj processing without certain parameters). The query parameter is used to perform any query against the specified DB-database (DB-obj), preferably a query that only returns a return code. Fig. 75B processing may return SELECT results in some embodiments (like find results returned). Preferably, the invoke command data is maintained to LBX History, a database log (e.g. incoming), or other useful storage for subsequent query use and a date/time stamp of when sent, and for DB query manager browse/use of the query in response to an applicable user action.
225	O	Invoking data causes modifying the value of the data at the particular MS (i.e. set data to value – like/see Notify data processing without certain parameters). An error can result if the data is not resolvable for the attempt. In the preferred embodiment, the data is a global system variable visible to all processes of a MS operating system. In other embodiments, the data may have limited scope which is made accessible to present disclosure processing (e.g. with extern). A recognized AppTerm causes access to record 5300 for proper semaphore synchronized access. Preferably, the data affected is maintained to LBX History, a historical log (e.g. incoming), or other useful storage for subsequent user browse, or programmatic access, of the data variable name, its before and after values & date/time stamp of sent, and for presentation of the data value for a user action to show it.

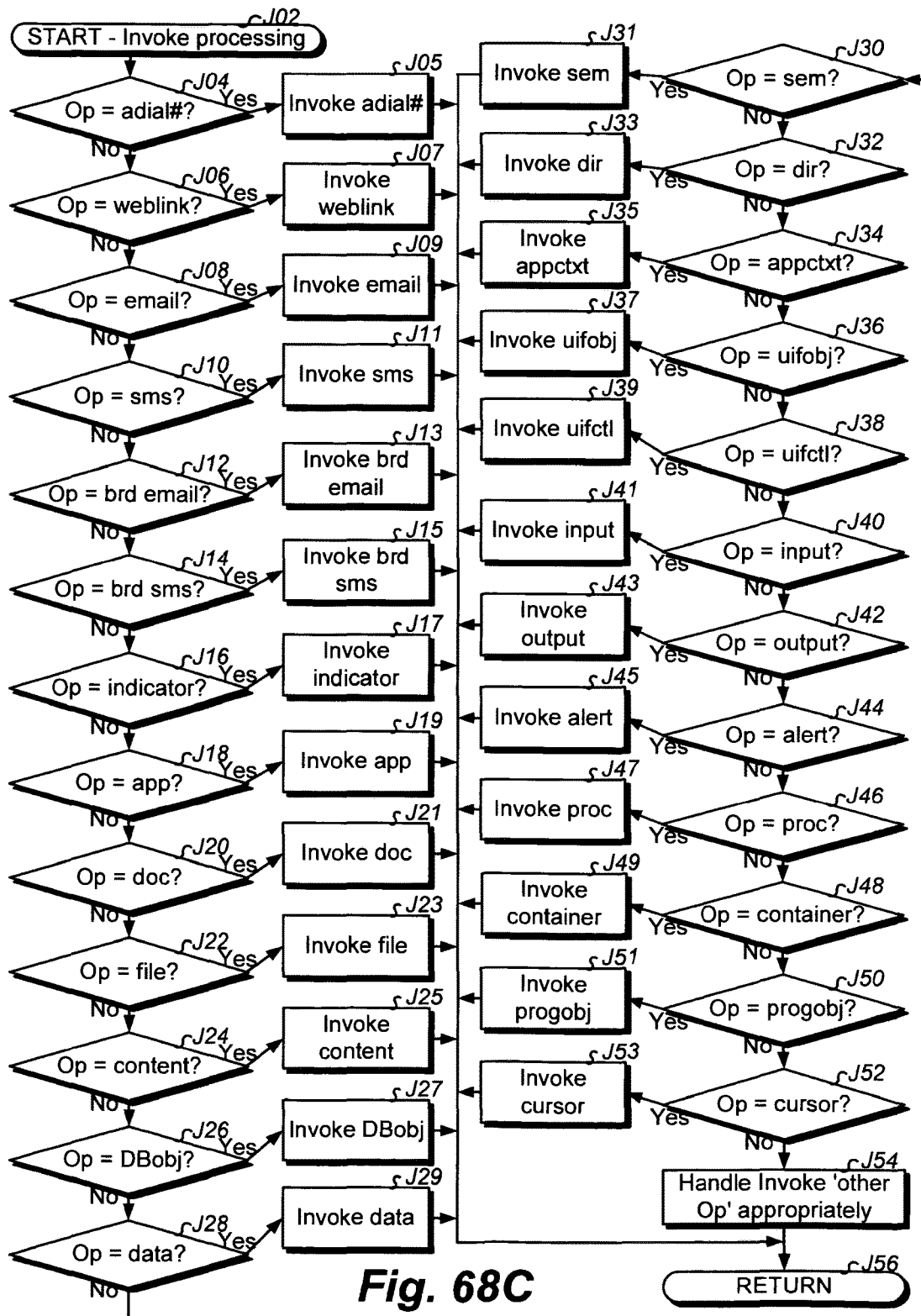
Fig. 68B-3

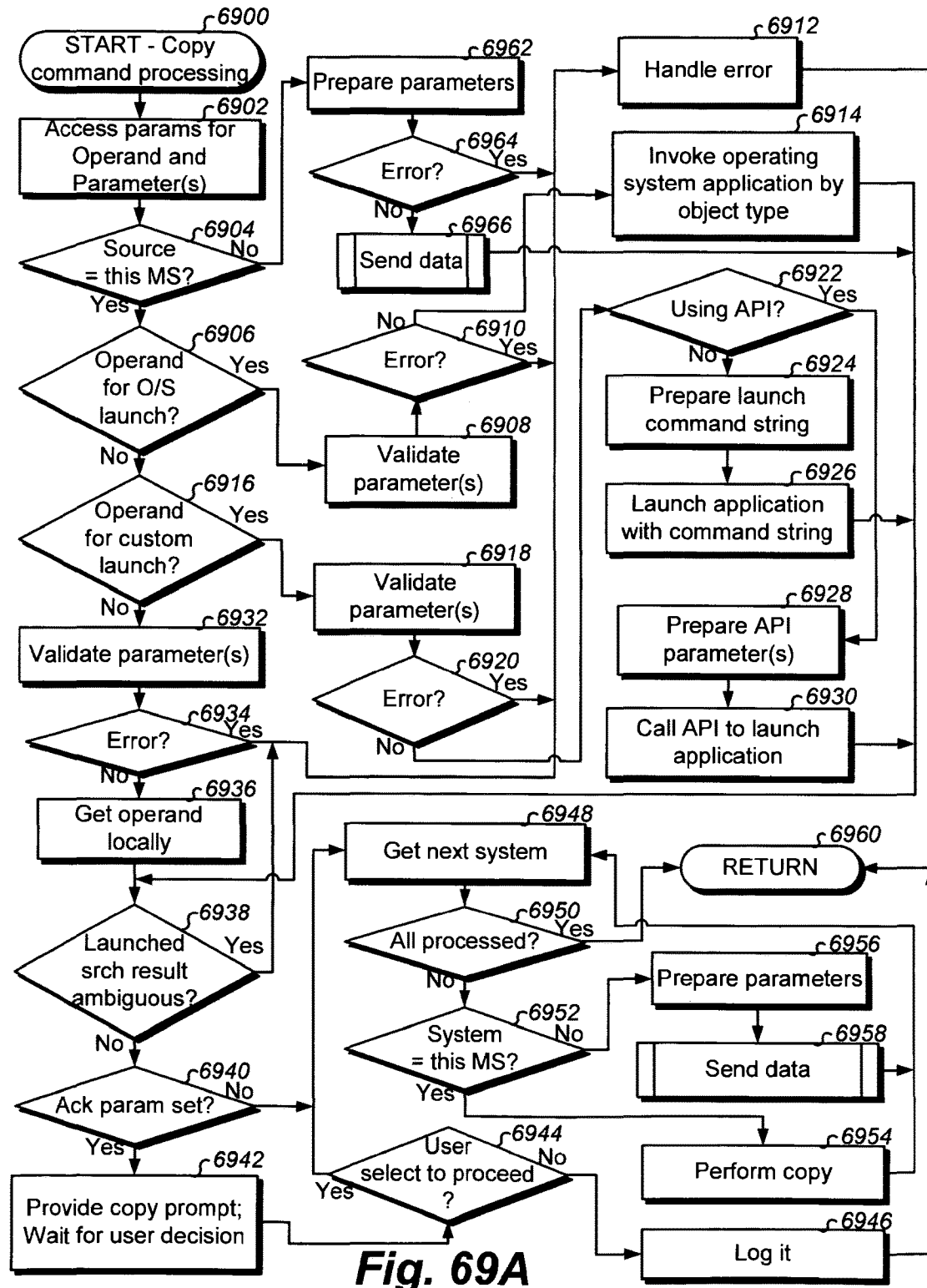
<i>Operand</i> ↓	PM	Preferred embodiment Invoke processing
227	O	Invoking a semaphore causes modifying the value of the semaphore at the particular MS where the action is being executed (like/see Notify semaphore processing). In the preferred embodiment, the semaphore is a global system semaphore visible to all processes of a MS operating system. In other embodiments, the semaphore may have limited scope which is made accessible to present disclosure processing (e.g. RAM semaphore). Preferably, the semaphore value before and after setting is maintained to LBX History, a historical log (e.g. incoming), or other useful storage for subsequent user browse, or programmatic access, and for presentation of the semaphore information in response to a user action to show it.
229	S	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
231	C	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
233	S	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter). Also, the cmds parameter replaces the "capture focused object" command string with one or more semicolon delimited "capture focused object" command strings for each target system in the system(s) parameters. This enables a plurality of different types of MSs to participate even though they have different commands (e.g. keystroke capture actions) to accomplish capturing the focused user interface object. Based on the file type at the particular MS, the appropriate application opens the file.
235	O	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
237	O	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
239	O	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
241	C	See Connect Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
243	O	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
245	S	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
247	O	See Notify Command for identical processing, except sender is forced to requesting MS, no documentary subj/msg parameter, and system(s) used instead of recipient(s). Processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).

Fig. 68B-4

<i>Operand</i> ↓	<u>PM</u>	<u>Preferred embodiment Invoke processing</u>
249	O	See Notify Command for identical processing, except sender is forced to requesting MS, no documentary subj/msg parameter, and system(s) used instead of recipient(s). Processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
251	O	See Send Command for identical processing, except sender is forced to requesting MS, and system(s) used instead of recipient(s) for calendar alteration without regard for the owner (API embodiment). Processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
253	O	See Send Command for identical processing, except sender is forced to requesting MS, and system(s) used instead of recipient(s) for AB alteration without regard for the owner (API embodiment). Processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
...		

Fig. 68B-5





<i>Operand</i>	PM	<u>Preferred embodiment Copy processing</u>
201	C	Copying an auto-dial # launches a phone number log interface with the auto-dial # parameter (can be wildcarded) for searching the source system. Preferably, both the outgoing and incoming logs are searched. In an alternate embodiment, the log is specified with a parameter. In the preferred embodiment, the most recent occurrence from a particular log is provided. In another embodiment, all occurrences found in history are presented with their date/time stamps, and perhaps other information, of the call and when it took place (e.g. when the ack parameter is set) and the user browses the results prior to accepting the copy of multiple items. The search takes place as though the user manually launched the search, entered the auto-dial # for the search, and then was provided with result(s) for the copy. Preferably, the copy shall take place if there are no ambiguities (e.g. more than one phone number returned per search criteria). An additional parameter may be specified for the target (different log) of the copy, otherwise the object is copied to an assumed location (e.g. same folder to more recent position). Appropriate MS storage is updated and subsequently processed as though the user manually performed the search and copy of the result. Preferably, the copy cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The copy is made to system(s) logs, preferably with identifying information of the source and who did the copy.
203	C	Copying a weblink launches a search to MS browser history with the weblink parameter (can be wildcarded) for searching the source system. In one embodiment, all occurrences found in history are presented with their date/time stamps, and perhaps other information, of the link and when it was invoked (e.g. when the ack parameter is specified to true) for presentation to the user prior to doing the copy. In the preferred embodiment, the most recent occurrence from a particular invocation is provided for the copy. An additional parameter may be specified for the target (specified favorites folder), otherwise the object is copied to an assumed location (e.g. highest level favorites folder or special named folder). The search takes place as though the user manually launched the search, entered the weblink for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy of the result. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The copy is made to a special browser favorites folder, or another designated folder configured ahead of time, preferably with identifying information of the source and who did the copy.

Fig. 69B-1

<i>Operand</i> ↓	PM	<u>Preferred embodiment Copy processing</u>
205	C	<p>Copying an email causes searching the source email system with search criteria of the email parameter string. The email parameter string can specify searching any email fields for any values including wildcarding (patterns for matching). Each field is referenced with a predefined name and then associated with a search criteria. For example, the email string of "subj:'personnel'; recip:'george@alltell.com'; body:'reduction in force'" causes searching all emails with a subject containing "personnel" and was sent to "george@alltell.com" and has a message body containing the string "reduction in force". To search for certain email containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:sent,inbox,company;" indicates to only search the email folders of sent, inbox, and company (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of email. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In one embodiment, all occurrences found in history are presented with at least their date/time stamps, subject line, sender and recipient, and perhaps other information, of the email and when it took place, when the ack parameter is set to true for user reconciliation. In a preferred embodiment, the most recent occurrence from searched folders is provided for the copy. An additional parameter may be specified for the target (specified folder), otherwise the object is copied to an assumed location (e.g. drafts, inbox, or special named folder). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for doing the copy. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The copy is made to a special email folder of the target system, or another designated folder configured ahead of time, or as specified with a new parameter for copy processing, preferably with identifying information of the source and who did the copy (if supported in email application).</p>

Fig. 69B-2

Operand ↓	PM	<u>Preferred embodiment Copy processing</u>
207	C	<p>Copying an sms message causes searching the source messaging system with search criteria of the sms message parameter string. The message parameter string can specify searching any message fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria. For example, the sms message string of "recip:'2144034071@nextel.com,9725397137@lbxsrv.com';" causes searching all messages to the sought recipients. To search for certain messaging containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:outgoing" indicates to only search the outgoing folder (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of messages. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In one embodiment, all occurrences found in history are presented with at least their date/time stamps, message, sender and recipient, and perhaps other information, of the message and when it took place, when the ack parameter is set to true for user reconciliation. In a preferred embodiment, the most recent occurrence from searched folders is provided for copying. An additional parameter may be specified for the target (specified folder), otherwise the object is copied to an assumed location (e.g. inbox, drafts, special named folder). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The copy is made to a special messaging folder of the target system, or another designated folder configured ahead of time, or as specified with a new parameter to copy processing, preferably with identifying information of the source and who did the copy.</p>

Fig. 69B-3

Operand ↓	<u>PM</u>	<u>Preferred embodiment Copy processing</u>
209	C	<p>Copying a broadcast email causes searching the source email system with search criteria of the email parameter string. The email parameter string can specify searching any email fields for any values including wildcarding (patterns for matching). Each field is referenced with a predefined name and then associated with a search criteria. For example, the email string of "subj:'personnel'; recip:'george@alltell.com'; body:'reduction in force'" causes searching all emails with a subject containing "personnel" and was sent to "george@alltell.com" and has a message body containing the string "reduction in force". To search for certain email containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:sent, inbox, company;" indicates to only search the email folders of sent, inbox, and company (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of email. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In one embodiment, all occurrences found in history are presented with at least their date/time stamps, subject line, sender and recipient, and perhaps other information, of the email and when it took place, when the ack parameter is set to true for user reconciliation. In a preferred embodiment, the most recent occurrence from searched folders is provided for the copy. An additional parameter may be specified for the target (specified folder), otherwise the object is copied to an assumed location (e.g. inbox, drafts, special named folder). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for doing the copy. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The copy is made to a special email folder of the target system, or another designated folder configured ahead of time, or as specified with a new parameter to copy processing, preferably with identifying information of the source and who did the copy.</p>

Fig. 69B-4

<i>Operand</i> ↓	PM	<u>Preferred embodiment Copy processing</u>
211	C	Copying a broadcast sms msg causes searching the source messaging system with search criteria of the sms message parameter string. The message parameter string can specify searching any message fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria. For example, the sms message string of "recip:'2144034071@nextel.com,9725397137@lboxsrv.com';" causes searching all messages to the sought recipients. To search for certain messaging containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:outgoing" indicates to only search the outgoing folder (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of messages. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In one embodiment, all occurrences found in history are presented with at least their date/time stamps, message, sender and recipient, and perhaps other information, of the message and when it took place, when the ack parameter is set to true for user reconciliation. In a preferred embodiment, the most recent occurrence from searched folders is provided for copying. An additional parameter may be specified for the target (specified folder), otherwise the object is copied to an assumed location (e.g. inbox, drafts, special named folder). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The copy is made to a special messaging folder of the target system, or another designated folder configured ahead of time, or as specified with a new parameter to copy processing, preferably with identifying information of the source and who did the copy.
213	C	Copying an indicator searches appropriate source storage for the indicator (e.g. storage/memory used for indicators by other commands). The indicator parameter string specifies the indicator string being sought and wildcarding is supported. In one embodiment, appropriate MS storage/memory which contains the history of indicators sent to the source system is searched and all occurrences found in history are presented with at least their date/time stamps, the indicator, and perhaps other information, for user reconciliation at block 6942. In a preferred embodiment, the most recently delivered indicator is identified and used for the copy. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for the copy. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The copy is made so that the target system(s) are delivered the indicator(s) like delivering a new indicator for presentation.

Fig. 69B-5

<i>Operand</i> ↓	PM	<u>Preferred embodiment Copy processing</u>
215	C	Copying an application causes searching the source system for the application (and with the params parameter(s) if specified to get the params specified invocation of the application). The app parameter is preferably an executable name and may contain parameters that were passed. Providing a more defined partial or full path to the application parameter will limit the search result. The app parameter string preferably supports wildcarding. In one embodiment, all occurrences found at the source and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information, for user reconciliation at block 6942. In a preferred embodiment, the most recently executed instance of the matching application is determined for the copy. In one embodiment, the application itself is copied to the target systems, perhaps as directed by an additional parameter (e.g. directory location). In another embodiment, the executable path to run the application is placed into execution history at the system(s) so that a user can run it, albeit from a remote system (assumption that application available for running there already). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copy. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
217	C	Copying a document causes searching the source for the document. The doc parameter is a document name. The document parameter can be a wildcard (pattern) for matching. Providing a more defined partial or full path to the document name will narrow the search. In one embodiment, all occurrences found on the MS and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information, for reconciliation at block 6942. In a preferred embodiment, the most recently accessed search result is provided for the copy. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Copying the document places a copy to each target system at a special shared folder, or configured folder for sharing, or as specified with a new destination parameter to copy processing.

Fig. 69B-6

Operand ↓	PM	<u>Preferred embodiment Copy processing</u>
219	C	Copying a file causes searching the source path for the file. The path parameter is a file name. Providing a more defined partial or full path to the file will narrow the search result. The path parameter can be a wildcard (pattern) for matching. In one embodiment, all occurrences found on the MS and their paths are provided with at least their date/time stamps, size, and perhaps attributes information, for reconciliation at block 6942. In a preferred embodiment, the most recently accessed file meeting the search criteria is copied to the target systems. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In one embodiment, the copy results in overwriting an existing file with the same handle (e.g. name). In another embodiment, the copy results in writing a newly altered name of the file when there is an existing file with the same handle (e.g. name). An additional parameter may be specified for the target (specified folder), otherwise the object is copied to an assumed location.
221	O	Copying content causes searching the source for the content. The content parameter is a reference to the content. The content parameter can be a wildcard (pattern) for matching. In a preferred embodiment, the most recently accessed search result is provided for the copy. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Copying the content places a copy to each target system at a special shared destination, or configured destination for sharing, or as specified with a new parameter to copy processing.

Fig. 69B-7

Operand ↓	PM	<u>Preferred embodiment Copy processing</u>
223	O	<p>Copying a DB object causes searching the source for the database object value. The database object parameter is provided with a variety of syntaxes depending on the type of database object sought. For example, the DB-obj parameters is "T:tablename" to seek a table, "S:schemaname" to seek a particular schema, "C:columnname" to seek a particular column name, "D:DBname" to seek a particular DB name, "R:rolename" to seek a particular role set, "P:procname" to search for particular stored procedure, etc. There are unique syntaxes for every type of DB object being sought which maps to an appropriate SQL system tables query. The search criteria can be a wildcard (pattern) for matching. In one embodiment, all occurrences found on the source system and information about the occurrence is presented to the user for reconciliation at block 6942. In other embodiments, the best (e.g. most recently accessed) fit database object is identified for use in the copy, or a new parameter indicating how to search. The search takes place as though the user manually launched the search, entered the criteria or query for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The value of the DB object is copied to the value of the DB object with the same name and type at the destination system(s). If not found at a target system, then no action is performed at that system. Copying a database object copies the value to the same database object(s) at other system(s), or creates new copies of the DB objects there when names do not match. Copying a DB object is intended to keep DBs in synch in some uses. Value(s) are overwritten. An additional parameter may be specified for the target of the copy (e.g. schema path which may include credentials for authentication).</p>

Fig. 69B-8

Operand ↓	PM	<u>Preferred embodiment Copy processing</u>
225	O	<p>Copying data causes searching the source system for the data. In the preferred embodiment, the data is a global system variable visible to all processes of a MS operating system. In other embodiments, the data may have limited scope which is made accessible to present disclosure processing (e.g. with extern). Depending on the embodiment, data may be that which is contained in a program data segment, stack segment, and/or extra segment. There can be unique syntaxes for specifying which type of data is being sought (e.g. "S:dataname" for data parameter). The search criteria can be a wildcard (pattern) for matching. In the preferred embodiment, a well known location of link symbol information files are consulted, and in another embodiment a new parameter specifies where to look, or which symbol file of information to use. A recognized AppTerm causes access to record 5300 for proper semaphore synchronized access. In one embodiment, all occurrences found at the source system and information about the occurrence including its current value is presented to the user for reconciliation at block 6942. In a preferred embodiment, the best data value (e.g. most recently accessed if more than one matches) is provided for the copy. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The value of the copied data is copied to the data with the same name and type at the destination system(s). If not found at a target system, then no action is performed at that system, or an error is provided. Copying a data object copies the value to the same data object(s) at other system(s). Copying is intended to keep data in synch between systems in some uses. Value(s) are overwritten. An additional parameter may be specified for the target data name of the copy.</p>
227	O	<p>Copying a semaphore causes reading the current value of the semaphore at the source where the copy command action is being executed and then copying the current value to the same semaphore names at the target system(s). The semaphore param can be a wildcard (pattern) for matching. In the preferred embodiment, the semaphore is a global system semaphore visible to all processes of a MS operating system. In other embodiments, the semaphore may have limited scope which is made accessible to present disclosure processing. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The value (set or cleared) of the copied semaphore is copied to the semaphore with the same name and type at the destination system(s). If not found at a target system, then no action is performed at that system, or an error is provided. Copying a semaphore copies the value to the same semaphore at other system(s). Copying a semaphore is intended to keep systems in synch in some uses. Value(s) are overwritten. An additional parameter may be specified for the target sem name of the copy.</p>

Fig. 69B-9

<u>Operand</u>	<u>PM</u>	<u>Preferred embodiment Copy processing</u>
229	C	Copying a directory causes searching the source system for the directory. The path parameter is a directory name. Providing a more defined partial or full path to the directory parameter will narrow the search result. The path parameter can be a wildcard (pattern) for matching. In one embodiment, all occurrences found on the MS and their paths are provided with at least their date/time stamps, size, and perhaps attributes information, for reconciliation at block 6942. In a preferred embodiment, the most recently accessed directory meeting the search criteria is copied to the target systems. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In one embodiment, the copy results in overwriting an existing directory and files therein. In another embodiment, the copy results in writing a newly altered name of directory contents when there is a conflict (e.g. existing entity with same name). In another embodiment, an additional target path parameter is provided for where to place the directory.
231	C	Operand 215 (application object) is treated identically to this Operand 231 (application context) this LBX release (same params currently).
233	C	Copying a focused user interface object causes capturing the currently focused user interface object using the first parameter (e.g. Alt-Prtscrn; can be changed with the param) string syntax for keystroke(s) to capture the image, and then copying the graphics file (file type in various embodiments) to a shared destination, or a configured destination at the target system(s) or as specified with a new parameter. The capture takes place as though the user manually performed the capture action, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the capture and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In one embodiment, the copy results in overwriting an existing file with the same handle (e.g. name), which will be seldom since the graphics file name preferably contains a date/time stamp portion. In another embodiment, the copy results in writing a newly altered name of the file when there is an existing file with the same handle (e.g. name). An additional parameter may be specified for the target of the copy. An additional parameter may be specified for the target format of the copy whereby a conversion is caused (e.g. JPG to TIFF).
235	C	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter). Also, the ack parameter provides a reconciliation option.
237	C	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter). Also, the ack parameter provides a reconciliation option.

Fig. 69B-10

<u>Operand</u> ↓	<u>PM</u>	<u>Preferred embodiment Copy processing</u>
239	C	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter). Also, the ack parameter provides a reconciliation option.
241	C	Copying an alert causes searching the source for the alert. The alert parameter is the same parameter used to generate an alert (e.g. using another command), and can be wildcarded. In one embodiment, all occurrences found on the MS which is associated to the alerter application in use at the MS, and which is used for other commands disclosed, are provided to the user for reconciliation at block 6942 with at least their date/time stamps, and perhaps other information. In other embodiments, the most recently generated alert matching the alert search criteria is used for copying, or the search occurs as specified with an additional parameter. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The copy is made so that the target system(s) are delivered the alert(s) like delivering a new alert to the systems.
243	C	Copying a process causes first finding all process names running at the source (e.g. MS) which contain the pname string parameter (e.g. in UNIX: "ps -ef grep pname"). In one embodiment, all occurrences found running at the MS are presented with interesting programmatic information such as when started, its size, etc for reconciliation at block 6942. In the preferred embodiment, one process running in the source system is to be found (i.e. >1 = ambiguous). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying. Results are useful statistics about the process which is running at the source. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, historical log, or other useful storage for subsequent use. Useful statistic(s) about the process (perhaps which statistics specified with an additional parameter) are copied to an appropriate destination of the target system(s) for informative purposes (e.g. a special log file). In another embodiment, the alerter process and/or indicator methodology can be used as the destination for the copy for alerting a user at the target system. In another embodiment, there is a new parameter for which end result the copy will have (informative destination, handled like alert, handled like indicator).

Fig. 69B-11

<i>Operand</i>	PM	<u>Preferred embodiment Copy processing</u>
245	C	Copying a container causes searching the source system for the container. The container parameter can be well defined to narrow the search result, and may be wildcarded (pattern) for matching. In one embodiment, all occurrences found on the MS and their references/handles are provided with at least their date/time stamps, size, and perhaps attributes information, for reconciliation at block 6942. In a preferred embodiment, the most recently accessed container meeting the search criteria is copied to the target systems. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In one embodiment, the copy results in overwriting an existing container. In another embodiment, the copy results in writing a newly altered reference/handle of the container when there is a conflict (e.g. existing entity with same name). An additional parameter may be specified for the target, otherwise the object is copied to an assumed location.
247	C	Copying a program object first causes searching the source for the program object. In the preferred embodiment, a unique syntax is used for which type of program object is being sought (similar to above). There can be unique syntaxes for specifying which type of program object is being sought (e.g. "S:dataname"). In one embodiment, all occurrences found on the MS and information about the occurrence including its current value is presented to the user for reconciliation at block 6942. In a preferred embodiment, one program object is to be found (e.g. >1 = ambiguous). In one embodiment, a well known location of link symbol information files are consulted, and in another embodiment a new parameter specifies where to look, or which symbol file of information to use. The search criteria can be a wildcard (pattern) for matching. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Useful statistic(s) about the program object (perhaps which statistics specified with an additional parameter) are copied to an appropriate destination of the target system(s) for informative purposes (e.g. a special log file). In another embodiment, the alerter process and/or indicator methodology can be used as the destination for the copy for alerting a user at the target system. In another embodiment, there is new parameter for which end result the copy will have (informative destination, handled like alert, handled like indicator). An alternate embodiment works like Operand 223 wherein copying is intended to keep program object(s) between systems in synch. Such embodiments require source and target system processing to have access to the object(s) (this may limit participating object(s)).

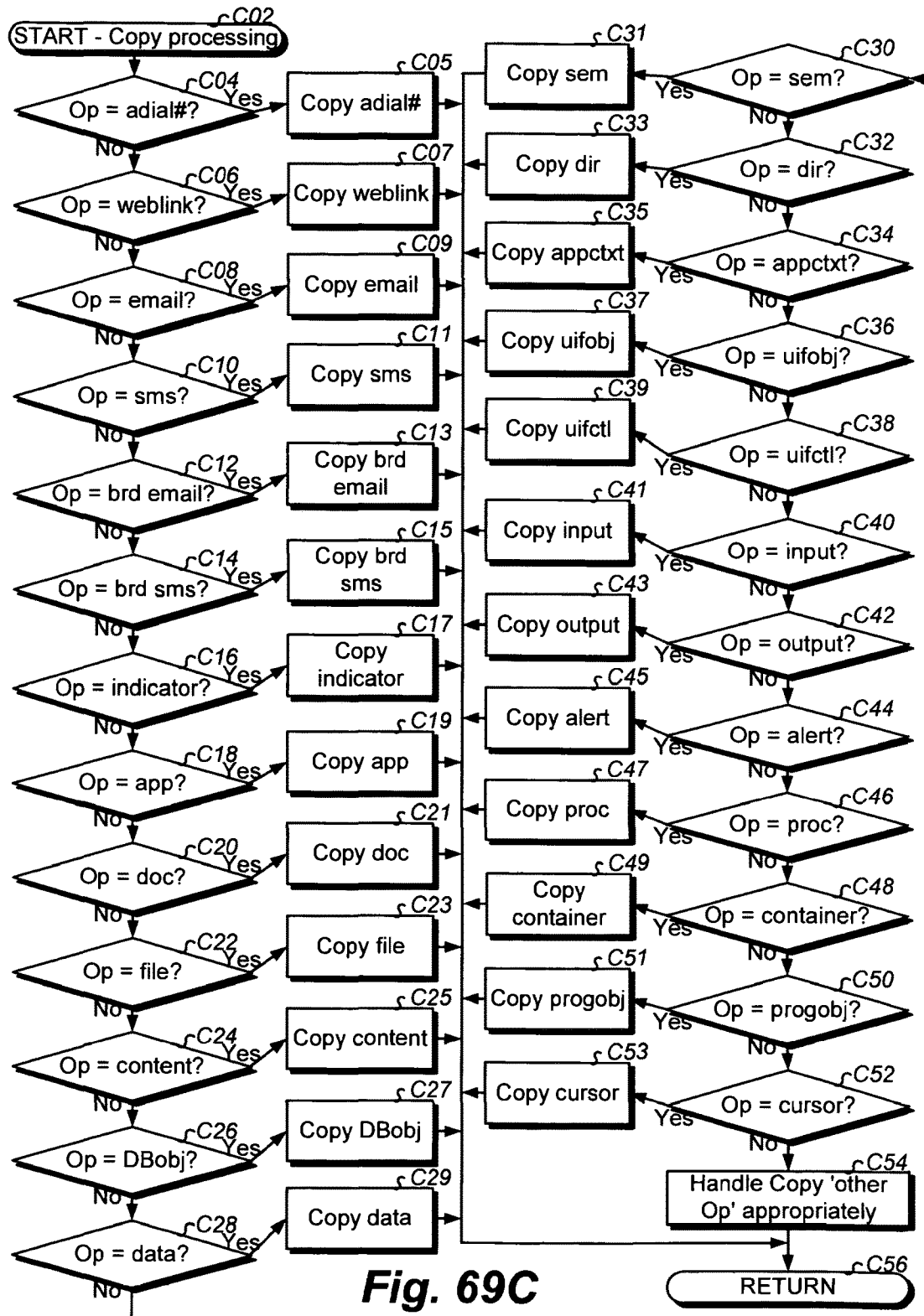
Fig. 69B-12

<i>Operand</i> ↓	PM	<u>Preferred embodiment Copy processing</u>
249	C	Copying a cursor causes searching the source system for the current cursor setting(s). In the preferred embodiment, provided in the search results is the current cursor information (e.g. image, animation , etc). The current cursor information of the source is then used to alter the cursor at the target system(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the copy operation. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
251	C	Copying a calendar (CAL) object causes searching the source CAL system with search criteria of the calendar object parameter string. The calendar object parameter string can specify searching any calendar entry fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria (similar to email above). Those skilled in the art recognize many useful syntaxes for searching any characteristics of CAL objects. In one embodiment, all occurrences found in history are presented to the user for reconciliation at block 6942 with at least their date/time stamps, attendees, and perhaps other information, of the CAL object and when it was scheduled. In a preferred embodiment, the most recent occurrence from the calendaring system is provided for the copy. Wildcarding (pattern matching) is preferably inherent by searching for substrings. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The calendar entry is copied in its entirety to the target system calendaring system. In another embodiment, a new parameter is specified to copy the calendar item to a new schedule or time. A duplicate calendar entry may be created if one already exists.

Fig. 69B-13

<i>Operand</i> ↓	<u>PM</u>	<u>Preferred embodiment Copy processing</u>
253	C	Copying an address book (AB) object causes searching the source AB system with search criteria of the AB object parameter string. The AB object parameter string can specify searching any AB entry fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria (similar to email above). Those skilled in the art recognize many useful syntaxes for searching any characteristics of AB objects/entries. In one embodiment, all occurrences found are presented to the user for reconciliation at block 6942 with appropriate AB information. In a preferred embodiment, the most recent occurrence from the AB system is provided for the copy. Wildcarding (pattern matching) is preferably inherent by searching for substrings. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The AB entry is copied in its entirety to the target system AB system. In another embodiment, a new parameter is specified to copy the AB item to special destination. A duplicate AB entry may be created if one already exists.
...		

Fig. 69B-14



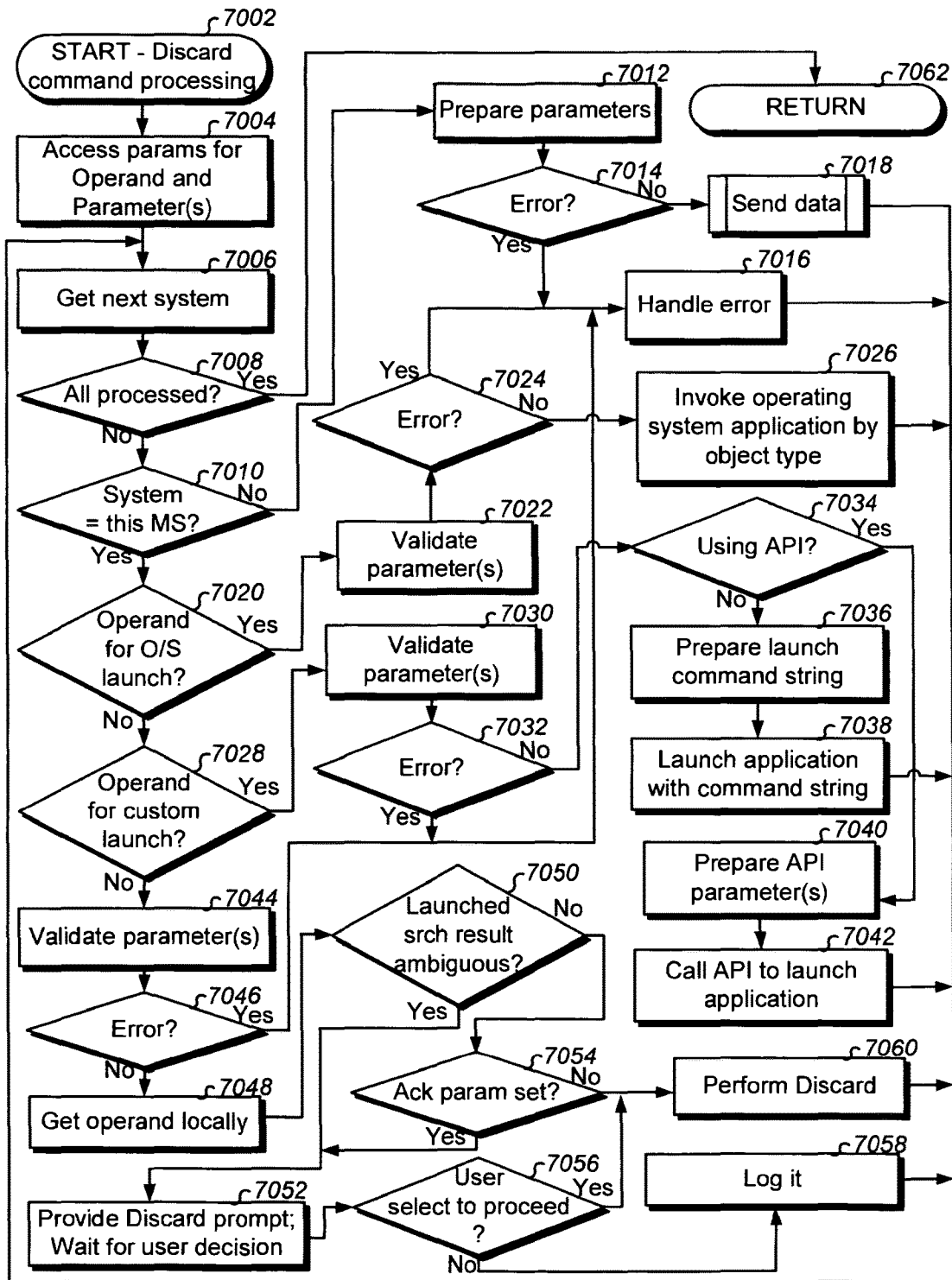


Fig. 70A

<i>Operand</i> ↓	PM	<u>Preferred embodiment Discard processing</u>
201	C	Discarding an auto-dial # launches a phone number log interface with the auto-dial # parameter (can be wildcarded) for searching the specified system (e.g. MS). Preferably, both the outgoing and incoming logs are searched. In an alternate embodiment, the log is specified with a parameter. In the preferred embodiment, the most recent occurrence from a particular log is to be discarded. In another embodiment, all occurrences found in history are presented with their date/time stamps, and perhaps other information, of the call and when it took place when the ack parameter is set and the user browses the results prior to accepting the discard of multiple items. The search takes place as though the user manually launched the search, entered the auto-dial # for the search, and then was provided with result(s) for the discard. Preferably, the discard shall take place if there are no ambiguities (e.g. more than one phone number returned per search criteria). Appropriate MS storage is updated and subsequently processed as though the user manually performed the search and discard of the result. Preferably, the discard cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The discard is made to system(s) logs.
203	S	Discarding a weblink launches a search to browser history with the weblink parameter (can be wildcarded) for searching the specified system. In one embodiment, all occurrences found in history are presented with their date/time stamps, and perhaps other information, of the link and when it was invoked, when the ack parameter is specified to true for presentation to the user prior to doing the discard. In the preferred embodiment, the most recent occurrence from a particular invocation is provided for the discard. The search takes place as though the user manually launched the search, entered the weblink for the search, and then was provided with the result for discarding it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard of the result. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The discard is made to a special browser favorites folder, another designated folder configured ahead of time, or as specified with an additional parameter.

Fig. 70B-1

<i>Operand</i> ↓	PM	<u>Preferred embodiment Discard processing</u>
205	C	<p>Discarding an email causes searching the specified email system with search criteria of the email parameter string. The email parameter string can specify searching any email fields for any values including wildcarding (patterns for matching). Each field is referenced with a predefined name and then associated with a search criteria. For example, the email string of "subj:'personnel'; recip:'george@alltell.com'; body:'reduction in force'" causes searching all emails with a subject containing "personnel" and was sent to "george@alltell.com" and has a message body containing the string "reduction in force". To search for certain email containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:sent,inbox,company;" indicates to only search the email folders of sent, inbox, and company (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of email. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In one embodiment, all occurrences found in history are presented with at least their date/time stamps, subject line, sender and recipient, and perhaps other information, of the email and when it took place, when the ack parameter is set to true for user reconciliation. In another embodiment, the most recent occurrence from searched folders is provided for the discard. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for doing the discard. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The email is discarded from an email folder as specified with a syntax in the email parameter string.</p>

Fig. 70B-2

Operand ↓	PM	<u>Preferred embodiment Discard processing</u>
207	C	<p>Discarding an sms message causes searching the specified messaging system with search criteria of the sms message parameter string. The message parameter string can specify searching any message fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria. For example, the sms message string of "recip:'2144034071@nextel.com,9725397137@lbxsrv.com';" causes searching all messages to the sought recipients. To search for certain messaging containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:outgoing" indicates to only search the outgoing folder (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of messages. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In one embodiment, all occurrences found in history are presented with at least their date/time stamps, message, sender and recipient, and perhaps other information, of the message and when it took place, when the ack parameter is set to true for user reconciliation. In another embodiment, the most recent occurrence from searched folders is provided for discarding. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for discarding it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The message is discarded from a folder as specified with a syntax in the sms message parameter string.</p>

Fig. 70B-3

<i>Operand</i> ↓	PM	<u>Preferred embodiment Discard processing</u>
209	C	<p>Discarding a broadcast email causes searching the specified email system with search criteria of the email parameter string. The email parameter string can specify searching any email fields for any values including wildcarding (patterns for matching). Each field is referenced with a predefined name and then associated with a search criteria. For example, the email string of "subj:'personnel'; recip:'george@alltell.com'; body:'reduction in force'" causes searching all emails with a subject containing "personnel" and was sent to "george@alltell.com" and has a message body containing the string "reduction in force". To search for certain email containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:sent,inbox,company;" indicates to only search the email folders of sent, inbox, and company (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of email. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In one embodiment, all occurrences found in history are presented with at least their date/time stamps, subject line, sender and recipient, and perhaps other information, of the email and when it took place, when the ack parameter is set to true for user reconciliation. In a preferred embodiment, the most recent occurrence from searched folders is provided for the discard. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for doing the discard. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The email is discarded from a folder as specified with a syntax in the email parameter string.</p>

Fig. 70B-4

<u>Operand</u>	<u>PM</u>	<u>Preferred embodiment Discard processing</u>
211	C	Discarding a broadcast sms msg causes searching the specified messaging system with search criteria of the sms message parameter string. The message parameter string can specify searching any message fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria. For example, the sms message string of "recip:'2144034071@nextel.com,9725397137@lbxsrv.com';" causes searching all messages to the sought recipients. To search for certain messaging containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:outgoing" indicates to only search the outgoing folder (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of messages. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In one embodiment, all occurrences found in history are presented with at least their date/time stamps, message, sender and recipient, and perhaps other information, of the message and when it took place, when the ack parameter is set to true for user reconciliation. In a preferred embodiment, the most recent occurrence from searched folders is provided for discarding. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for discarding it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The message is discarded from a folder as specified with a syntax in the sms message parameter string.
213	O	Discarding an indicator searches appropriate specified system storage for the indicator (e.g. storage/memory used for indicators by other commands). The indicator parameter string specifies the indicator string being sought and wildcarding is supported. In one embodiment, appropriate MS storage/memory which contains the history of indicators sent to the source system is searched and all occurrences found in history are presented with at least their date/time stamps, the indicator, and perhaps other information, for user reconciliation. In one embodiment, the most recently delivered indicator is identified and used for the discard. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for the discard. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The discard is performed so that the target system(s) have the indicator(s) removed from the interface if currently presented and removed from history maintained for the user interface object presentation (preferably not from LBX history). An additional parameter may specify how to delete the indicator.

Fig. 70B-5

<i>Operand</i> ↓	PM	Preferred embodiment Discard processing
215	C	Discarding an application causes searching the specified system for the application (and with the params parameter(s) if specified to get the right invocation of the application). The app parameter is preferably an executable name. Providing a partial or full path to the application parameter will limit the search result. The app parameter string preferably supports wildcarding. In one embodiment, all occurrences found at the source and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information, for user reconciliation. In a preferred embodiment, the most recently executed instance of the matching application is determined for the discard. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provide with the result for discard. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The discard does not remove the application from the target system. It terminates the application by terminating/killing it at the operating system level. A Discard File operand command can be used to remove it from the system.
217	S	Discarding a document causes searching the specified system for the document. The doc parameter is a document name. The document parameter can be a wildcard (pattern) for matching. Providing a more defined partial or full path to the document name will narrow the search. In one embodiment, all occurrences found and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information, for user reconciliation. In a preferred embodiment, the most recently accessed search result is provided for the discard. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for discarding it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Discard of the document removes it from each target system at a special shared folder, or configured folder for sharing, or as specified with a new parameter to discard processing.
219	S	Discarding a file causes searching the source path for the file. The path parameter is a file name. Providing a more defined partial or full path to the file will narrow the search result. The path parameter can be a wildcard (pattern) for matching. In one embodiment, all occurrences found on the MS and their paths are provided with at least their date/time stamps, size, and perhaps attributes information, for user reconciliation. In another embodiment, the most recently accessed file meeting the search criteria is discarded. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provide with the result for discarding it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. File(s) are discarded at each target system.

Fig. 70B-6

<i>Operand</i>	PM	<u>Preferred embodiment Discard processing</u>
221	O	Discarding content causes searching the specified system for the content. The content parameter is a reference to the content. The content parameter can be a wildcard (pattern) for matching. In a preferred embodiment, the most recently accessed search result is provided for the discard. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for discarding it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Discarding the content removes it from each target system at a special shared destination, or configured destination for sharing, or as specified with a new parameter to discard processing.
223	C	Discarding a DB object causes searching the specified system for the database object value. The database object parameter is provided with a variety of syntaxes depending on the type of database object sought. For example, the DB-obj parameters is "T:tablename" to seek a table, "S:schemaname" to seek a particular schema, "C:columnname" to seek a particular column name, "D:DBname" to seek a particular DB name, "R:rolename" to seek a particular role set, "P:procname" to search for particular stored procedure, etc. There are unique syntaxes for every type of DB object being sought. The search criteria can be a wildcard (pattern) for matching. In one embodiment, all occurrences found on the source system and information about the occurrence is presented to the user for reconciliation. In a preferred embodiment, the best (e.g. most recently accessed) fit database object is identified for discard. The search takes place as though the user manually launched the search, entered the criteria or query for the search, and then was provided with the result for discarding it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The DB object is discarded (removed, deleted, dropped).
225	O	Same as Compose processing except modifies the value to an initial value (e.g. 0) at each system.
227	O	Same as Compose processing except modifies the value to an initial value (e.g. clear) at each system.

Fig. 70B-7

<u>Operand</u>	<u>PM</u>	<u>Preferred embodiment Discard processing</u>
229	S	Discarding a directory causes searching the specified system for the directory. The path parameter is a directory name. Providing a more defined partial or full path to the directory parameter will narrow the search result. The path parameter can be a wildcard (pattern) for matching. In one embodiment, all occurrences found on the MS and their paths are provided with at least their date/time stamps, size, and perhaps attributes information, for user reconciliation. In one embodiment, the most recently accessed directory meeting the search criteria is discarded. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for discarding it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The directory is discarded at each target system.
231	C	Operand 215 and 235 (application object) is treated identically to Operand 231 (application context) this LBX release (same params currently). The specified application is terminated, not removed.
233	S	Discarding a user interface object causes closing/terminating the focused object(s) at each specified system that contains the objtxt parameter criteria in the titlebar. In a preferred embodiment, there is a unique syntax for which places of user interface objects that are currently active are to be search (e.g. title bar, entry fields, radio button options, window text, combinations thereof, etc). The search criteria can be a wildcard (pattern) for matching. The search takes place as though the user manually launched the search, entered the criteria for the search, and the object(s) are closed/terminated. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
235	C	Operand 215 and 231 (application object) is treated identically to Operand 235 (application context) this LBX release (same params currently). The specified application is terminated, not removed.
237	O	Discarding input causes reinitializing the iodev parameter input device stream of the specified system, so that any pending state is discarded. In one embodiment, a special input datastream is issued to reinitialize the I/O path. In another embodiment, the I/O path is terminated and restarted to reinitialize for the attached device(s). In another embodiment, the I/O path is flushed and then reinitialized. In another embodiment, an additional parameter indicates how to discard the iodev device stream (e.g. method and/or initialization data to initialize with). The iodev parameter specifies which Input/Output device to reinitialize. Preferably, the discard command data is maintained to LBX History, a log, or other useful storage for subsequent use.

Fig. 70B-8

<i>Operand</i> ↓	PM	<u>Preferred embodiment Discard processing</u>
239	O	Discarding output causes reinitializing the iodev parameter output device stream of the specified system, so that any pending state is discarded. In one embodiment, a special output datastream is issued to reinitialize the I/O path. In another embodiment, the I/O path is terminated and restarted to reinitialize for the attached device(s). In another embodiment, the I/O path is flushed and then reinitialized. In another embodiment, an additional parameter indicates how to discard the iodev device stream (e.g. method and/or initialization data to initialize with). The iodev parameter specifies which Input/Output device to reinitialize. Preferably, the discard command data is maintained to LBX History, a log, or other useful storage for subsequent use.
241	C	Discarding an alert causes searching the specified system for the alert and discarding it. The alert parameter is the same parameter used to generate an alert (e.g. using another command), and can be wildcarded. In one embodiment, all occurrences found which is associated to the alerter application in use at the MS, and which is used for other commands disclosed, are provided to the user for reconciliation with at least their date/time stamps, and perhaps other information. In one embodiment, the most recently generated alert matching the alert search criteria is used for discarding. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for discarding it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
243	O	Discarding a process causes searching for and terminating/killing all process names running at the specified system (e.g. MS) which contain the pname string parameter (e.g. in UNIX: "ps -ef grep pname"). In one embodiment, all occurrences found running at the MS are presented with interesting programmatic information such as when started, its size, etc for user reconciliation. In the preferred embodiment, one process running in the specified system is to be found (i.e. >1 = ambiguous). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for discarding. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, historical log, or other useful storage for subsequent use. The process is terminated/killed. It is not removed from the system.

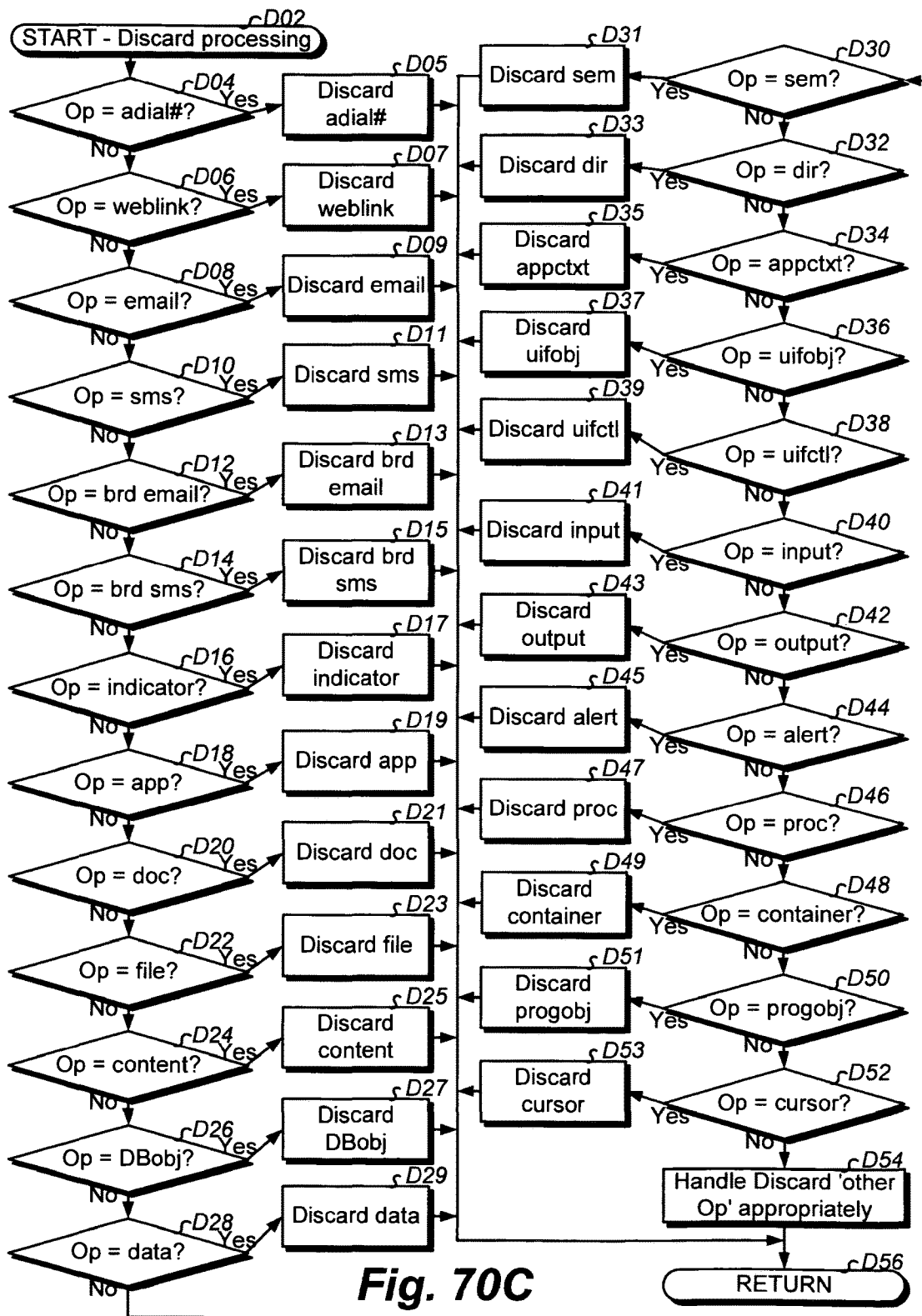
Fig. 70B-9

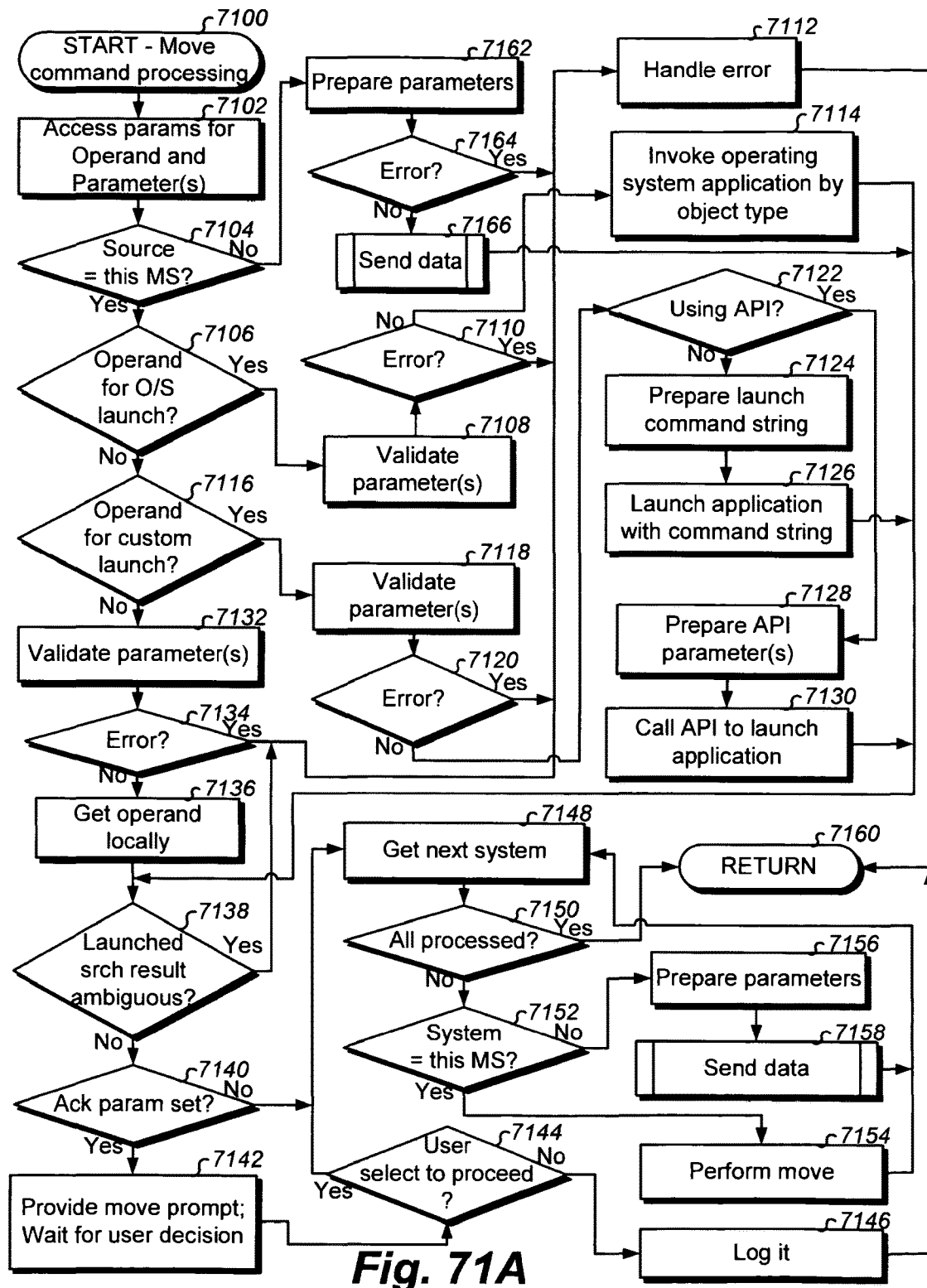
<i>Operand</i>	PM	Preferred embodiment Discard processing
245	S	Discarding a container causes searching the specified system for the container. The container parameter can be well defined to narrow the search result, and may be wildcarded (pattern) for matching. In one embodiment, all occurrences found on the MS and their references/handles are provided with at least their date/time stamps, size, and perhaps attributes information, for user reconciliation. In one embodiment, the most recently accessed container meeting the search criteria is discarded at the target systems. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for discarding it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The container is discarded from each specified system.
247	O	Discarding a program object causes searching the specified system for the program object and initializing to a initial value (i.e. discarding any current value). In the preferred embodiment, a unique syntax is used for which type of program object is being sought (similar to above). There can be unique syntaxes for specifying which type of program object is being sought (e.g. "S:dataname"). In one embodiment, all occurrences found on the MS and information about the occurrence including its current value is presented to the user for reconciliation. In a preferred embodiment, one program object is to be found (e.g. >1 = ambiguous). In one embodiment, a well known location of link symbol information files are consulted, and in another embodiment a new parameter specifies where to look, or which symbol file of information to use. The search criteria can be a wildcard (pattern) for matching. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for discarding. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. A reasonable reset value (e.g. 0 for data, clear for semaphore) is set to the program object. Program objects which cannot hold a value (procedure) are preferably not affected by the discard command. Local and remote processing must have programmatic visibility to affected program object(s).
249	O	Discarding a cursor causes resetting the cursor at the specified system. In the preferred embodiment, provided in the search results is the current cursor information (e.g. image, animation , etc) when user reconciliation is involved. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the discard operation. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In one embodiment, a system defaulted cursor is set. In another embodiment, the previous cursor setting is returned to, and multiple discard cursor actions can change the cursor continuously to historical settings. An additional parameter may provide how to discard the cursor.

Fig. 70B-10

<i>Operand</i> ↓	PM	Preferred embodiment Discard processing
251	C	Discarding a calendar object causes searching the specified calendar system with search criteria of the calendar object parameter string. The calendar object parameter string can specify searching any calendar entry fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria (similar to email above). Those skilled in the art recognize many useful syntaxes for searching any characteristics of calendar objects. In one embodiment, all occurrences found in history are presented to the user for reconciliation with at least their date/time stamps, sender and recipient, and perhaps other information, of the calendar object and when it was scheduled. In a preferred embodiment, the most recent occurrence from the calendaring system is discarded. Wildcarding (pattern matching) is preferably inherent by searching for substrings. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for discarding it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The calendar entry(s) are discarded from the target system calendaring system.
253	C	Discarding an address book (AB) object causes searching the specified AB system with search criteria of the AB object parameter string. The AB object parameter string can specify searching any AB entry fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria (similar to email above). Those skilled in the art recognize many useful syntaxes for searching any characteristics of AB objects/entries. In one embodiment, all occurrences found are presented to the user for reconciliation with appropriate AB information. In a preferred embodiment, the most recent occurrence from the AB system is discarded. Wildcarding (pattern matching) is preferably inherent by searching for substrings. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for discarding it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and discard. Preferably, the discard command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The AB entry(s) are discarded from the target system AB system.
...		

Fig. 70B-11





<i>Operand</i> ↓	PM	<u>Preferred embodiment Move processing</u>
201	C	Moving an auto-dial # launches a phone number log interface with the auto-dial # parameter (can be wildcarded) for searching the source system. Preferably, both the outgoing and incoming logs are searched. In an alternate embodiment, the log is specified with a parameter. In the preferred embodiment, the most recent occurrence from a particular log is provided. In another embodiment, all occurrences found in history are presented with their date/time stamps, and perhaps other information, of the call and when it took place (e.g. when the ack parameter is set) and the user browses the results prior to accepting the move. The search takes place as though the user manually launched the search, entered the auto-dial # for the search, and then was provided with result(s) for the move. Preferably, the move shall take place if there are no ambiguities (e.g. more than one phone number returned per search criteria). An additional parameter may be specified for the target (different log) of the move, otherwise the object is moved to an assumed location (e.g. same folder to more recent position). Appropriate MS storage is updated and subsequently processed as though the user manually performed the search and move of the result. Preferably, the move cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The move is made to system(s) logs, preferably with identifying information of the source and who did the move.
203	C	Moving a weblink launches a search to MS browser history with the weblink parameter (can be wildcarded) for searching the source system. In one embodiment, all occurrences found in history are presented with their date/time stamps, and perhaps other information, of the link and when it was invoked (e.g. when the ack parameter is specified to true) for presentation to the user prior to doing the move. In the preferred embodiment, the most recent occurrence from a particular invocation is provided for the move. An additional parameter may be specified for the target (specified favorites folder), otherwise the object is moved to an assumed location (e.g. highest level favorites folder). The search takes place as though the user manually launched the search, entered the weblink for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move of the result. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The move is made to a special browser favorites folder, or another designated folder configured ahead of time, or as specified with an additional parameter, preferably with identifying information of the source and who did the move.

Fig. 71B-1

<i>Operand</i> ↓	PM	<u>Preferred embodiment Move processing</u>
205	C	<p>Moving an email causes searching the source email system with search criteria of the email parameter string. The email parameter string can specify searching any email fields for any values including wildcarding (patterns for matching). Each field is referenced with a predefined name and then associated with a search criteria. For example, the email string of "subj:'personnel'; recip:'george@alltell.com'; body:'reduction in force'" causes searching all emails with a subject containing "personnel" and was sent to "george@alltell.com" and has a message body containing the string "reduction in force". To search for certain email containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:sent,inbox,company;" indicates to only search the email folders of sent, inbox, and company (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of email. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In one embodiment, all occurrences found in history are presented with at least their date/time stamps, subject line, sender and recipient, and perhaps other information, of the email and when it took place, when the ack parameter is set to true for user reconciliation. In a preferred embodiment, the most recent occurrence from searched folders is provided for the move. An additional parameter may be specified for the target (specified folder), otherwise the object is moved to an assumed location (e.g. inbox, drafts, special named folder). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for doing the move. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The move is made to a special email folder of the target system, or another designated folder configured ahead of time, or as specified with a new parameter for move processing, preferably with identifying information of the source and who did the move (if supported in email application).</p>

Fig. 71B-2

Operand ↓	PM	<u>Preferred embodiment Move processing</u>
207	C	<p>Moving an sms message causes searching the source messaging system with search criteria of the sms message parameter string. The message parameter string can specify searching any message fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria. For example, the sms message string of "recip:'2144034071@nextel.com,9725397137@lbxsrv.com';" causes searching all messages to the sought recipients. To search for certain messaging containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:outgoing" indicates to only search the outgoing folder (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of messages. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In one embodiment, all occurrences found in history are presented with at least their date/time stamps, message, sender and recipient, and perhaps other information, of the message and when it took place, when the ack parameter is set to true for user reconciliation. In a preferred embodiment, the most recent occurrence from searched folders is provided for moving. An additional parameter may be specified for the target (specified folder), otherwise the object is moved to an assumed location (e.g. inbox, drafts, special named folder). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The move is made to a special messaging folder of the target system, or another designated folder configured ahead of time, or as specified with a new parameter to move processing, preferably with identifying information of the source and who did the move.</p>

Fig. 71B-3

<u>Operand</u> ↓	<u>PM</u>	<u>Preferred embodiment Move processing</u>
209	C	<p>Moving a broadcast email causes searching the source email system with search criteria of the email parameter string. The email parameter string can specify searching any email fields for any values including wildcarding (patterns for matching). Each field is referenced with a predefined name and then associated with a search criteria. For example, the email string of "subj:'personnel';recip:'george@alltell.com';body:'reduction in force'" causes searching all emails with a subject containing "personnel" and was sent to "george@alltell.com" and has a message body containing the string "reduction in force". To search for certain email containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:sent,inbox,company;" indicates to only search the email folders of sent, inbox, and company (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of email. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In one embodiment, all occurrences found in history are presented with at least their date/time stamps, subject line, sender and recipient, and perhaps other information, of the email and when it took place, when the ack parameter is set to true for user reconciliation. In a preferred embodiment, the most recent occurrence from searched folders is provided for the move. An additional parameter may be specified for the target (specified folder), otherwise the object is moved to an assumed location (e.g. inbox, drafts, special named folder). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for doing the move. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The move is made to a special email folder of the target system, or another designated folder configured ahead of time, or as specified with a new parameter to move processing, preferably with identifying information of the source and who did the move.</p>

Fig. 71B-4

<i>Operand</i> ↓	PM	<u>Preferred embodiment Move processing</u>
211	C	Moving a broadcast sms msg causes searching the source messaging system with search criteria of the sms message parameter string. The message parameter string can specify searching any message fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria. For example, the sms message string of "recip:'2144034071@nextel.com,9725397137@lboxsrv.com';" causes searching all messages to the sought recipients. To search for certain messaging containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:outgoing" indicates to only search the outgoing folder (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of messages. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In one embodiment, all occurrences found in history are presented with at least their date/time stamps, message, sender and recipient, and perhaps other information, of the message and when it took place, when the ack parameter is set to true for user reconciliation. In a preferred embodiment, the most recent occurrence from searched folders is provided for moving. An additional parameter may be specified for the target (specified folder), otherwise the object is moved to an assumed location (e.g. inbox, drafts, special named folder). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The move is made to a special messaging folder of the target system, or another designated folder configured ahead of time, or as specified with a new parameter to move processing, preferably with identifying information of the source and who did the move.
213	C	Moving an indicator searches appropriate source storage for the indicator (e.g. storage/memory used for indicators by other commands). The indicator parameter string specifies the indicator string being sought and wildcarding is supported. In one embodiment, appropriate MS storage/memory which contains the history of indicators sent to the source system is searched and all occurrences found in history are presented with at least their date/time stamps, the indicator, and perhaps other information, for user reconciliation. In a preferred embodiment, the most recently delivered indicator is identified and used for the move. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for the move. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The move is made so that the target system(s) are delivered the indicators like delivering new indicator(s) for presentation.

Fig. 71B-5

Operand ↓	PM	<u>Preferred embodiment Move processing</u>
215	C	Moving an application causes searching the source system for the application (and with the params parameter(s) if specified to get the param specified invocation of the application). The app parameter is preferably an executable name, and may contain parameters that were passed. Providing a more defined partial or full path to the application parameter will limit the search result. The app parameter string preferably supports wildcarding. In one embodiment, all occurrences found at the source and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information, for user reconciliation. In a preferred embodiment, the most recently executed instance of the matching application is determined for the move. In one embodiment, the application itself is moved to the target systems, perhaps as directed by an additional parameter (e.g. directory location). In another embodiment, the executable path to run the application is moved to execution history at the system(s) so that a user can run it, albeit from a remote system (assumption that application available for running there already). In another embodiment, the executable(s) are moved to the target system using methodologies of U.S. Patent 5,938,722 ("Method of executing programs in a network", Johnson). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for move. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
217	C	Moving a document causes searching the source system for the document. The doc parameter is a document name. The document parameter can be a wildcard (pattern) for matching. Providing a more defined partial or full path to the document name will narrow the search. In one embodiment, all occurrences found on the MS and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information, for user reconciliation. In a preferred embodiment, the most recently accessed search result is provided for the move. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Moving the document places the document to each target system at a special shared folder, or configured folder for sharing, or as specified with a new destination parameter to move processing.

Fig. 71B-6

<i>Operand</i> ↓	PM	Preferred embodiment Move processing
219	C	Moving a file causes searching the source path for the file. The path parameter is a file name. Providing a more defined partial or full path to the file will narrow the search result. The path parameter can be a wildcard (pattern) for matching. In one embodiment, all occurrences found on the MS and their paths are provided with at least their date/time stamps, size, and perhaps attributes information, for user reconciliation. In a preferred embodiment, the most recently accessed file meeting the search criteria is moved to the target systems. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In one embodiment, the move results in overwriting an existing file with the same handle (e.g. name). In another embodiment, the move may result in writing a newly altered name of the file when there is an existing file with the same handle (e.g. name). An additional parameter may be specified for the target (specified folder), otherwise the object is moved to an assumed location.
221	O	Moving content causes searching the source for the content. The content parameter is a reference to the content. The content parameter can be a wildcard (pattern) for matching. In a preferred embodiment, the most recently accessed search result is provided for the move. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Moving the content places the content to each target system at a special shared destination, or configured destination for sharing, or as specified with a new parameter to move processing.

Fig. 71B-7

Operand ↓	<u>PM</u>	<u>Preferred embodiment Move processing</u>
223	O	<p>Moving a DB object causes searching the source for the database object value. The database object parameter is provided with a variety of syntaxes depending on the type of database object sought. For example, the DB-obj parameters is "T:tablename" to seek a table, "S:schemaname" to seek a particular schema, "C:columnname" to seek a particular column name, "D:DBname" to seek a particular DB name, "R:rolename" to seek a particular role set, "P:procname" to search for particular stored procedure, etc. There are unique syntaxes for every type of DB object being sought which maps to an appropriate SQL system tables query. The search criteria can be a wildcard (pattern) for matching. In one embodiment, all occurrences found on the source system and information about the occurrence is presented to the user for reconciliation. In other embodiments, the best (e.g. most recently accessed) fit database object is identified for use in the move, or a new parameter indicates how to search. The search takes place as though the user manually launched the search, entered the criteria or query for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The value of the DB object is moved to the value of the DB object with the same name and type at the destination system(s). If not found at a target system, then no action is performed at that system. Moving a database object moves the value to the same database object(s) at other system(s), or creates new ones when there is not match. Value(s) are overwritten. An additional parameter may be specified for the target of the move.</p>

Fig. 71B-8

Operand ↓	<u>PM</u>	<u>Preferred embodiment Move processing</u>
225	O	<p>Moving data causes searching the source system for the data. In the preferred embodiment, the data is a global system variable visible to all processes of a MS operating system. In other embodiments, the data may have limited scope which is made accessible to present disclosure processing (e.g. with extern). Depending on the embodiment, data may be that which is contained in a program data segment, stack segment, and/or extra segment. There can be unique syntaxes for specifying which type of data is being sought (e.g. "S:dataname" for data parameter). The search criteria can be a wildcard (pattern) for matching. In the preferred embodiment, a well known location of link symbol information files are consulted, and in another embodiment a new parameter specifies where to look, or which symbol file of information to use. A recognized AppTerm causes access to record 5300 for proper semaphore synchronized access. In one embodiment, all occurrences found at the source system and information about the occurrence including its current value is presented to the user for reconciliation. In a preferred embodiment, the best data value (e.g. most recently accessed if more than one matches) is provided for the move. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The value of the data to be moved is copied to the data with the same name and type at the destination system(s). In another embodiment, the source data is reinitialized (e.g. 0), or reinitialized according to a new parameter, as part of the move operation. If not found at a target system, then no action is performed at that system, or an error is provided. Moving a data object at least copies the value to the same data object(s) at other system(s). Value(s) are overwritten. An additional parameter may be specified for the target of the move.</p>

Fig. 71B-9

<u>Operand</u>	<u>PM</u>	<u>Preferred embodiment Move processing</u>
227	O	Moving a semaphore causes reading the current value of the semaphore at the source where the move command action is being executed and then moving the current value to the same semaphore names at the target system(s). The semaphore param can be a wildcard (pattern) for matching. In the preferred embodiment, the semaphore is a global system semaphore visible to all processes of a MS operating system. In other embodiments, the semaphore may have limited scope which is made accessible to present disclosure processing. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The value (set or cleared) of the moved semaphore is copied to the semaphore with the same name and type at the destination system(s). In another embodiment, the source semaphore is reinitialized (e.g. clear), or reinitialized according to a new parameter, as part of the move operation. If not found at a target system, then no action is performed at that system, or an error is provided. Moving a semaphore at least copies the value to the same semaphore at other system(s). Value(s) are changed (clear or set). An additional parameter may be specified for the target of the move.
229	C	Moving a directory causes searching the source system for the directory. The path parameter is a directory name. Providing a more defined partial or full path to the directory parameter will narrow the search result. The path parameter can be a wildcard (pattern) for matching. In one embodiment, all occurrences found on the MS and their paths are provided with at least their date/time stamps, size, and perhaps attributes information, for user reconciliation. In a preferred embodiment, the most recently accessed directory meeting the search criteria is copied to the target systems. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In one embodiment, the move results in overwriting an existing directory and files therein. In another embodiment, the move results in writing a newly altered name of directory contents when there is a conflict (e.g. existing entity with same name). In another embodiment, an additional target path parameter is provided for where to place the directory.
231	C	Operand 215 (application object) is treated identically to this Operand 231 (application context) this LBX release (same params currently).

Fig. 71B-10

<i>Operand</i> ↓	PM	<u>Preferred embodiment Move processing</u>
233	C	Moving a focused user interface object causes capturing the currently focused user interface object using the first parameter (e.g. Alt-Prtsrn; can be changed with the param) string syntax for keystroke(s) to capture the image, and then moving the graphics file (file type in various embodiments) to a shared destination, or a configured destination at the target system(s), or as specified with a new parameter. The capture takes place as though the user manually performed the capture action, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the capture and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In one embodiment, the move results in overwriting an existing file with the same handle (e.g. name), which will be seldom since the graphics file name preferably contains a date/time stamp portion. In another embodiment, the move results in writing a newly altered name of the file when there is an existing file with the same handle (e.g. name). An additional parameter may be specified for the target of the move. An additional parameter may be specified for the target format of the move whereby a conversion is caused (e.g. JPG to TIFF).
235	C	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter). Also, the ack parameter provides a reconciliation option.
237	C	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter). Also, the ack parameter provides a reconciliation option.
239	C	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter). Also, the ack parameter provides a reconciliation option.
241	C	Moving an alert causes searching the source system for the alert. The alert parameter is the same parameter used to generate an alert (e.g. using another command), and can be wildcarded. In one embodiment, all occurrences found on the MS which is associated to the alerter application in use at the MS, and which is used for other commands disclosed, are provided to the user for reconciliation with at least their date/time stamps, and perhaps other information. In other embodiments, the most recently generated alert matching the alert search criteria is used for moving, or the search occurs as specified with an additional parameter. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The move is made so that the target system(s) are delivered the alert(s) like delivering a new alert to the systems.

Fig. 71B-11

<i>Operand</i>	<u>PM</u>	<u>Preferred embodiment Move processing</u>
243	C	Moving a process causes first finding all process names running at the source (e.g. MS) which contain the pname string parameter (e.g. in UNIX: "ps -ef grep pname"). In one embodiment, all occurrences found running at the MS are presented with interesting programmatic information such as when started, its size, etc for user reconciliation. In the preferred embodiment, one process running in the source system is to be found (i.e. >1 = ambiguous). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving. Results are useful statistics about the process which is running at the source. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, historical log, or other useful storage for subsequent use. Useful statistic(s) about the process (perhaps which statistics specified with an additional parameter) are moved/copied to an appropriate destination of the target system(s) for informative purposes (e.g. a special log file). In another embodiment, the alerter process and/or indicator methodology can be used as the destination for the move for alerting a user at the target system. In another embodiment, there is a new parameter for which end result the move will have (informative destination, handled like alert, handled like indicator). In some embodiments, the process is roved to the target system using methodologies of U.S. Patent 5,938,722 ("Method of executing programs in a network", Johnson), as requested in a new parameter.
245	C	Moving a container causes searching the source system for the container. The container parameter can be well defined to narrow the search result, and may be wildcarded (pattern) for matching. In one embodiment, all occurrences found on the MS and their references/handles are provided with at least their date/time stamps, size, and perhaps attributes information, for user reconciliation. In a preferred embodiment, the most recently accessed container meeting the search criteria is copied to the target systems. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In one embodiment, the move results in overwriting an existing container. In another embodiment, the move results in writing a newly altered reference/handle of the container when there is a conflict (e.g. existing entity with same name). An additional parameter may be specified for the target, otherwise the object is moved to an assumed location.

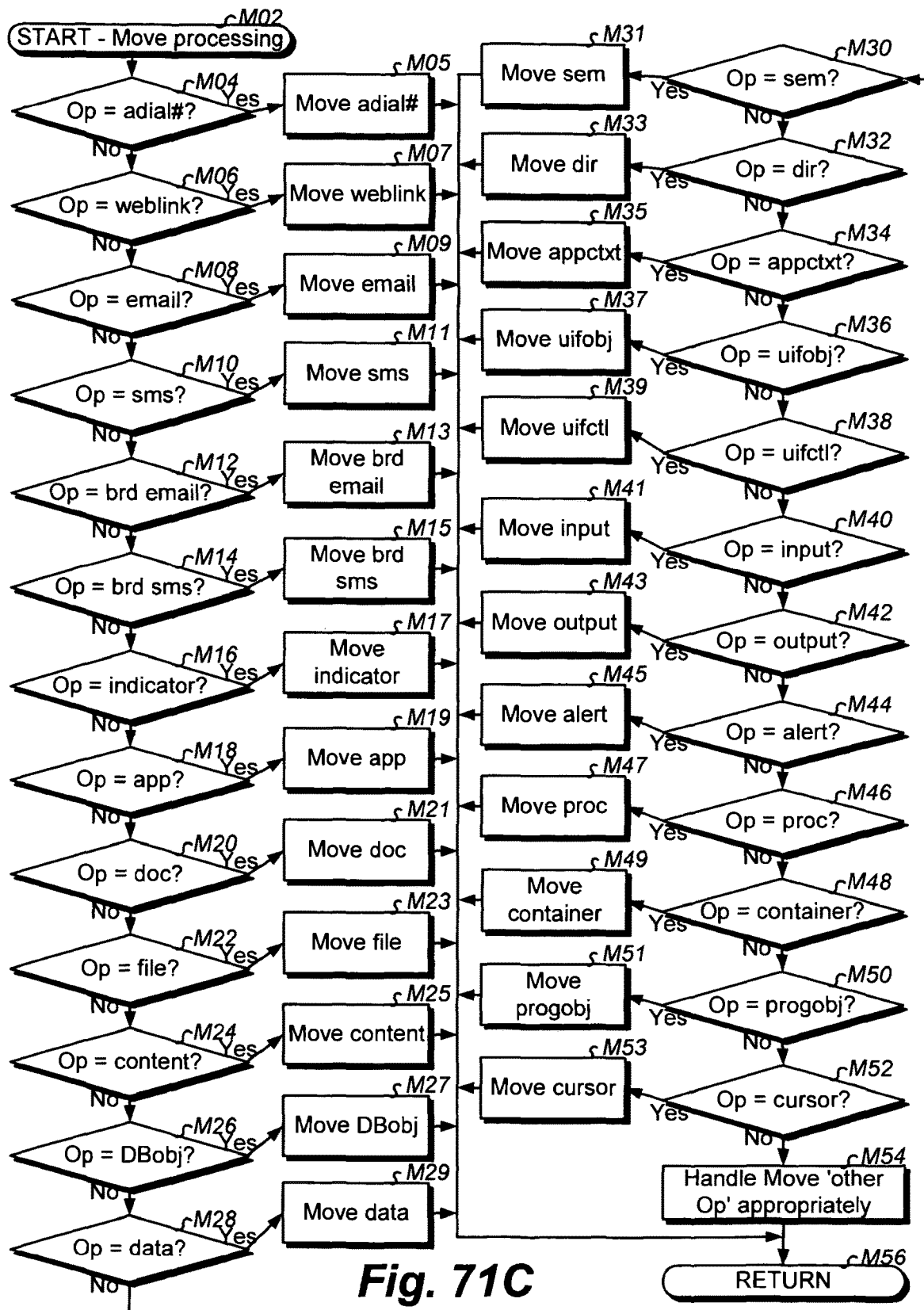
Fig. 71B-12

<i>Operand</i> ↓	PM	<u>Preferred embodiment Move processing</u>
247	C	<p>Moving a program object causes first searching the source for the program object. In the preferred embodiment, a unique syntax is used for which type of program object is being sought (similar to above). There can be unique syntaxes for specifying which type of program object is being sought (e.g. "S:dataname"). In one embodiment, all occurrences found on the MS and information about the occurrence including its current value is presented to the user for reconciliation. In a preferred embodiment, one program object is to be found (e.g. >1 = ambiguous). In one embodiment, a well known location of link symbol information files are consulted, and in another embodiment a new parameter specifies where to look, or which symbol file of information to use. The search criteria can be a wildcard (pattern) for matching. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Useful statistic(s) about the program object (perhaps which statistics specified with an additional parameter) are moved/copied to an appropriate destination of the target system(s) for informative purposes (e.g. a special log file). In another embodiment, the alerter process and/or indicator methodology can be used as the destination for the move for alerting a user at the target system. In another embodiment, there is new parameter for which end result the move will have (informative destination, handled like alert, handled like indicator). An alternate embodiment works like Operand 223 wherein moving is intended to keep program object(s) between systems in synch, albeit with a discard from the source system. Such embodiments require source and target system processing to have access to the object(s) (this may limit participating object(s)).</p>
249	C	<p>Moving a cursor causes searching the source system for the current cursor setting(s). In the preferred embodiment, provided in the search results is the current cursor information (e.g. image, animation, etc). The current cursor information of the source is then used to alter the cursor at the target system(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the move operation. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In some embodiments, the source cursor is reset to a different (e.g. initialized) setting as resulting from the move.</p>

Fig. 71B-13

Operand ↓	PM	<u>Preferred embodiment Move processing</u>
251	C	Moving a calendar object causes searching the source calendar system with search criteria of the calendar object parameter string. The calendar object parameter string can specify searching any calendar entry fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria (similar to email above). Those skilled in the art recognize many useful syntaxes for searching any characteristics of calendar objects. In one embodiment, all occurrences found in history are presented to the user for reconciliation with at least their date/time stamps, attendees, and perhaps other information, of the calendar object and when it was scheduled. In a preferred embodiment, the most recent occurrence from the calendaring system is provided for the move. Wildcarding (pattern matching) is preferably inherent by searching for substrings. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The calendar entry is moved in its entirety to the target system calendaring system. In another embodiment, a new parameter is specified to move the calendar item(s) to a new schedule or time. A duplicate calendar entry may be created if one already exists.
253	C	Moving an address book (AB) object causes searching the source AB system with search criteria of the AB object parameter string. The AB object parameter string can specify searching any AB entry fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria (similar to email above). Those skilled in the art recognize many useful syntaxes for searching any characteristics of AB objects/entries. In one embodiment, all occurrences found are presented to the user for reconciliation with appropriate AB information. In a preferred embodiment, the most recent occurrence from the AB system is provided for the move. Wildcarding (pattern matching) is preferably inherent by searching for substrings. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for moving it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and move. Preferably, the move command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The AB entry is moved in its entirety to the target system AB system. In another embodiment, a new parameter is specified to move the AB item(s) to a special destination. A duplicate AB entry may be created if one already exists.
...		

Fig. 71B-14



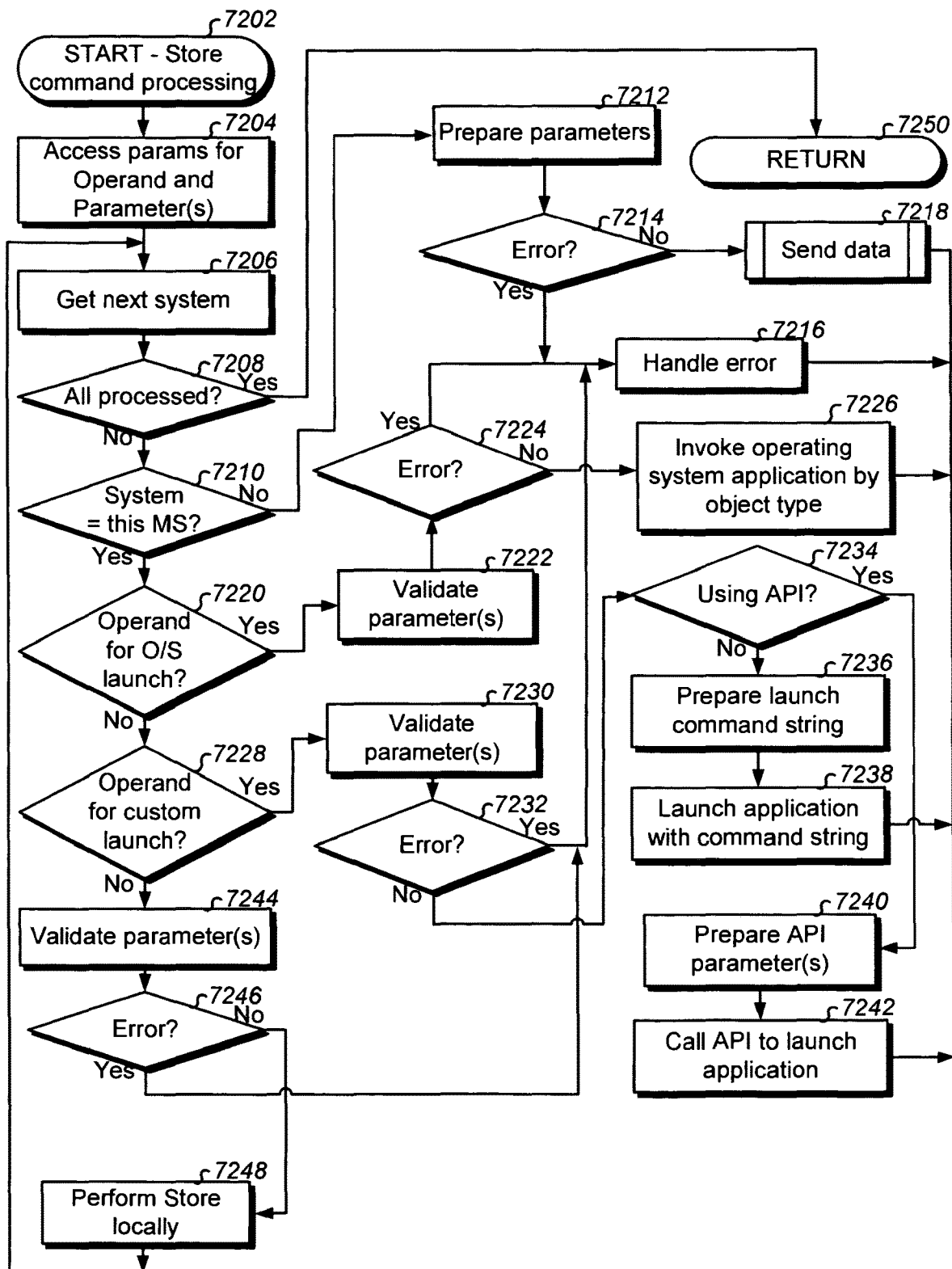


Fig. 72A

<i>Operand</i> ↓	PM	<u>Preferred embodiment Store processing</u>
201	C	Storing an auto-dial # stores the auto-dial # parameter to the system(s). Preferably, a certain log is used to store the auto-dial #, or an additional parameter can be provided for where to store the auto-dial # to. Store processing takes place as though the user manually launched it. Appropriate MS storage is updated and subsequently processed as though the user manually performed the store command. Preferably, the store cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In a preferred embodiment, information about who, when, why is additionally maintained with the stored auto-dial #. An additional parameter may be specified for how/where exactly to store it (e.g. which log).
203	S	Storing a weblink stores the weblink parameter to the system(s). Preferably, a certain link folder is used to store it, or an additional parameter can be provided for where to store it. Store processing takes place as though the user manually launched it. Appropriate MS storage is updated and subsequently processed as though the user manually performed the store command. Preferably, the store cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In a preferred embodiment, information about who, when, why is additionally maintained with the stored weblink. An additional parameter may be specified for how/where exactly to store it (e.g. which folder).
205	C	Storing an email causes storing the email object to the system(s)' email system. In one embodiment, the email parameter string is a string containing a syntax for defining an email item and used to create the email to a certain folder, configured folder, or as specified with an additional parameter. Each email field can be defined with values, and the store command may result in defaulting fields which are not specified in the email parameter. In another embodiment, the email parameter points to a file containing a syntax for creating the email object. Those skilled in the art recognize many useful syntaxes for setting data in a new email object. Store processing takes place as though the user manually launched it. Appropriate MS storage is updated and subsequently processed as though the user manually performed the store command. Preferably, the store cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In a preferred embodiment, information about who, when, why is additionally maintained with the stored email.

Fig. 72B-1

<i>Operand</i> ↓	<u>PM</u>	<u>Preferred embodiment Store processing</u>
207	C	Storing an sms message causes storing the sms message object to the system(s)' messaging system. In one embodiment, the sms message parameter string is a string containing a syntax for defining an sms message item and used to create the sms message to a certain folder, configured folder, or as specified with an additional parameter. Each sms message field can be defined with values, and the store command may result in defaulting fields which are not specified in the sms message parameter. In another embodiment, the sms message parameter points (e.g. path) to a file containing a syntax for creating the sms message object. Those skilled in the art recognize many useful syntaxes for setting data in a new sms message object. Store processing takes place as though the user manually launched it. Appropriate MS storage is updated and subsequently processed as though the user manually performed the store command. Preferably, the store cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In a preferred embodiment, information about who, when, why is additionally maintained with the stored sms message.
209	C	Storing a broadcast email causes storing the email object to the system(s)' email system. In one embodiment, the email parameter string is a string containing a syntax for defining an email item and used to create the email to a certain folder, configured folder, or as specified with an additional parameter. Each email field can be defined with values, and the store command may result in defaulting fields which are not specified in the email parameter. In another embodiment, the email parameter points (e.g. path) to a file containing a syntax for creating the email object. Those skilled in the art recognize many useful syntaxes for setting data in a new email object. Store processing takes place as though the user manually launched it. Appropriate MS storage is updated and subsequently processed as though the user manually performed the store command. Preferably, the store cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In a preferred embodiment, information about who, when, why is additionally maintained with the stored email.

Fig. 72B-2

<i>Operand</i> ↓	PM	<u>Preferred embodiment Store processing</u>
211	C	Storing a broadcast sms message causes storing the sms message object to the system(s)' messaging system. In one embodiment, the sms message parameter string is a string containing a syntax for defining an sms message item and used to create the sms message to a certain folder, configured folder, or as specified with an additional parameter. Each sms message field can be defined with values, and the store command may result in defaulting fields which are not specified in the sms message parameter. In another embodiment, the sms message parameter points to a file (e.g. path) containing a syntax for creating the sms message object. Those skilled in the art recognize many useful syntaxes for setting data in a new sms message object. Store processing takes place as though the user manually launched it. Appropriate MS storage is updated and subsequently processed as though the user manually performed the store command. Preferably, the store cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In a preferred embodiment, information about who, when, why is additionally maintained with the stored sms message.
213	O	See Invoke Command for identical processing.
215	C	See Invoke Command for identical processing.
217	S	Storing a document stores the document parameter to the system(s). The document may be a self contained object parameter, or pointer (e.g. path) to a file defining the document. Preferably, a certain folder is used to store it, or an additional parameter can be provided for where to store it. Store processing takes place as though the user manually launched it. Appropriate MS storage is updated and subsequently processed as though the user manually performed the store command. Preferably, the store cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In a preferred embodiment, information about who, when, why is additionally maintained with the stored document.
219	S	Storing a file stores the file from the path parameter to the system(s). Preferably, a certain folder is used to store it, or an additional parameter can be provided for where to store it. Store processing takes place as though the user manually launched it. Appropriate MS storage is updated and subsequently processed as though the user manually performed the store command. Preferably, the store cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In a preferred embodiment, information about who, when, why is additionally maintained with the stored file.

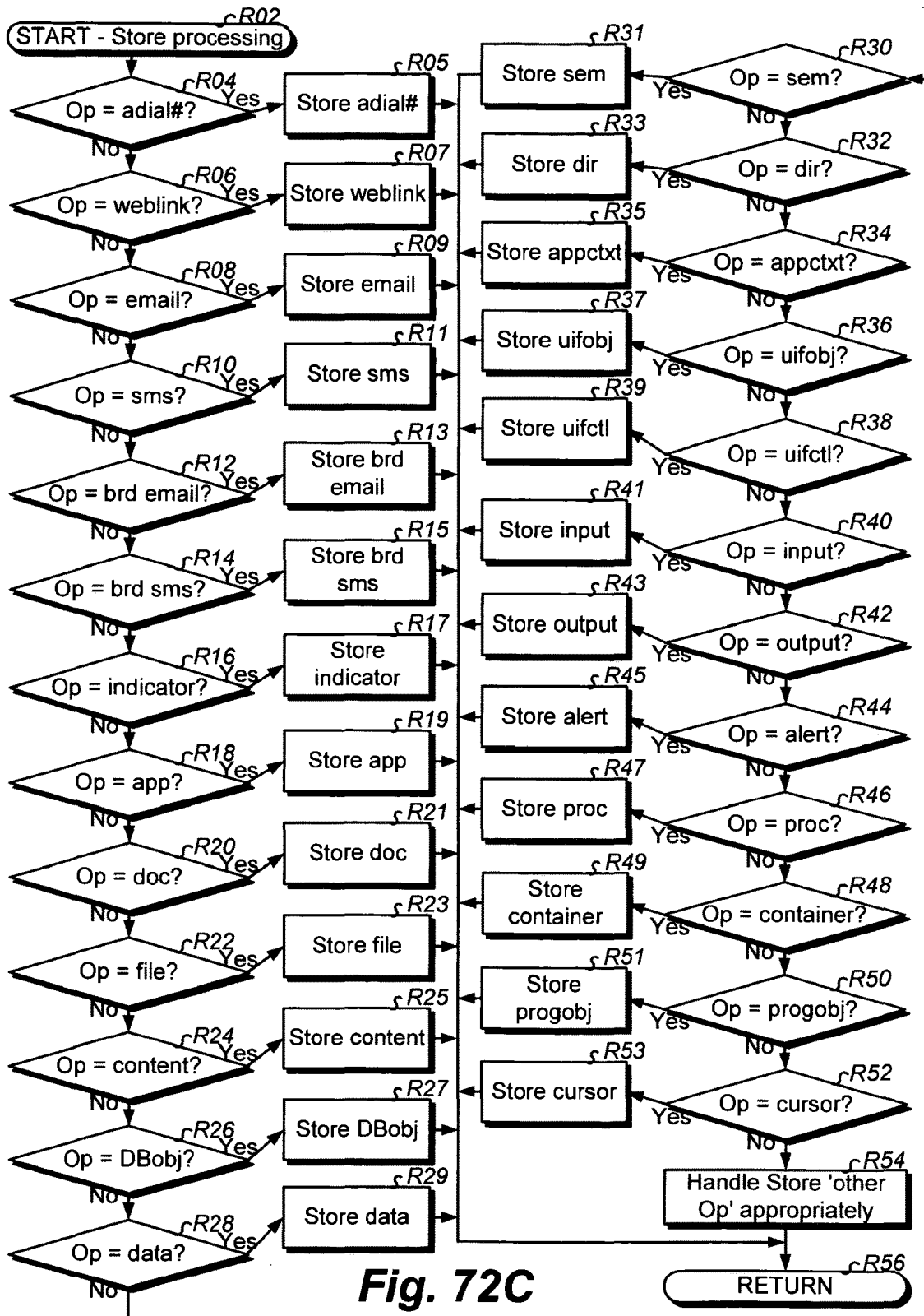
Fig. 72B-3

<i>Operand</i>	PM	Preferred embodiment Store processing
221	O	Storing content stores the content parameter to the system(s). The content may be a self contained object parameter, or pointer (e.g. path) to a file defining the content. Preferably, a certain destination is used to store it, or an additional parameter can be provided for where to store it. Store processing takes place as though the user manually launched it. Appropriate MS storage is updated and subsequently processed as though the user manually performed the store command. Preferably, the store cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In a preferred embodiment, information about who, when, why is additionally maintained with the stored content.
223	C	See Invoke Command for identical processing.
225	O	See Invoke Command for identical processing.
227	O	See Invoke Command for identical processing.
229	S	Storing a directory stores the directory from the path parameter to the system(s). Preferably, a certain folder is used to store it, or an additional parameter can be provided for where to store it. Store processing takes place as though the user manually launched it. Appropriate MS storage is updated and subsequently processed as though the user manually performed the store command. Preferably, the store cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In a preferred embodiment, information about who, when, why is additionally maintained with the stored directory.
231	C	See Invoke Command for identical processing.
233	S	Storing a focused user interface object causes a snapshot to be taken of the currently focused user interface object (to .jpg, .gif, alternate embodiments, etc) at the MS and then the snapshot file is stored as though the user manually captured the focused user interface object (e.g. Alt-Prtsrn) and saved it. Preferably, a certain folder is used to store it, or an additional parameter can be provided for where to store it. The first parameter command syntax can be defaulted or changed. Appropriate MS storage is updated and subsequently processed as though the user manually performed the store command. Preferably, the store cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In a preferred embodiment, information about who, when, why is additionally maintained with the stored snapshot. An additional parameter may be specified for the target format of the store whereby a conversion is caused (e.g. JPG to TIFF).
235	O	See Invoke Command for identical processing.
237	O	See Invoke Command for identical processing.
239	O	See Invoke Command for identical processing.

Fig. 72B-4

<i>Operand</i> ↓	PM	Preferred embodiment Store processing
241	S	Storing an alert causes storing the alert to the specified system(s). The alert parameter is the same parameter used to generate an alert (e.g. using another command). Storing takes place as though the user manually launched it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the store. Preferably, the store command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The store is performed so that the target system(s) are delivered the alert(s) like delivering a new alert to the systems.
243	O	Storing a process causes sending an operating system signal (see UNIX signaling) to the process name (after determining the Process ID (PID) of the pname parameter). A numeric value parameter (e.g. 0 or 1) may be communicated with the signal. An error is logged if the process is not found for signaling. Preferably, the store command data is maintained to LBX History, a log, or other useful storage for subsequent use.
245	S	Storing a container stores the container parameter to the system(s). The container may be a self contained object parameter, or pointer (e.g. path) to a file defining the container. Preferably, a certain destination is used to store it, or an additional parameter can be provided for where to store it. Store processing takes place as though the user manually launched it. Appropriate MS storage is updated and subsequently processed as though the user manually performed the store command. Preferably, the store cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. In a preferred embodiment, information about who, when, why is additionally maintained with the stored container.
247	O	See Invoke Command for identical processing.
249	O	See Invoke Command for identical processing.
251	C	See Invoke Command for identical processing.
253	C	See Invoke Command for identical processing.
...		

Fig. 72B-5



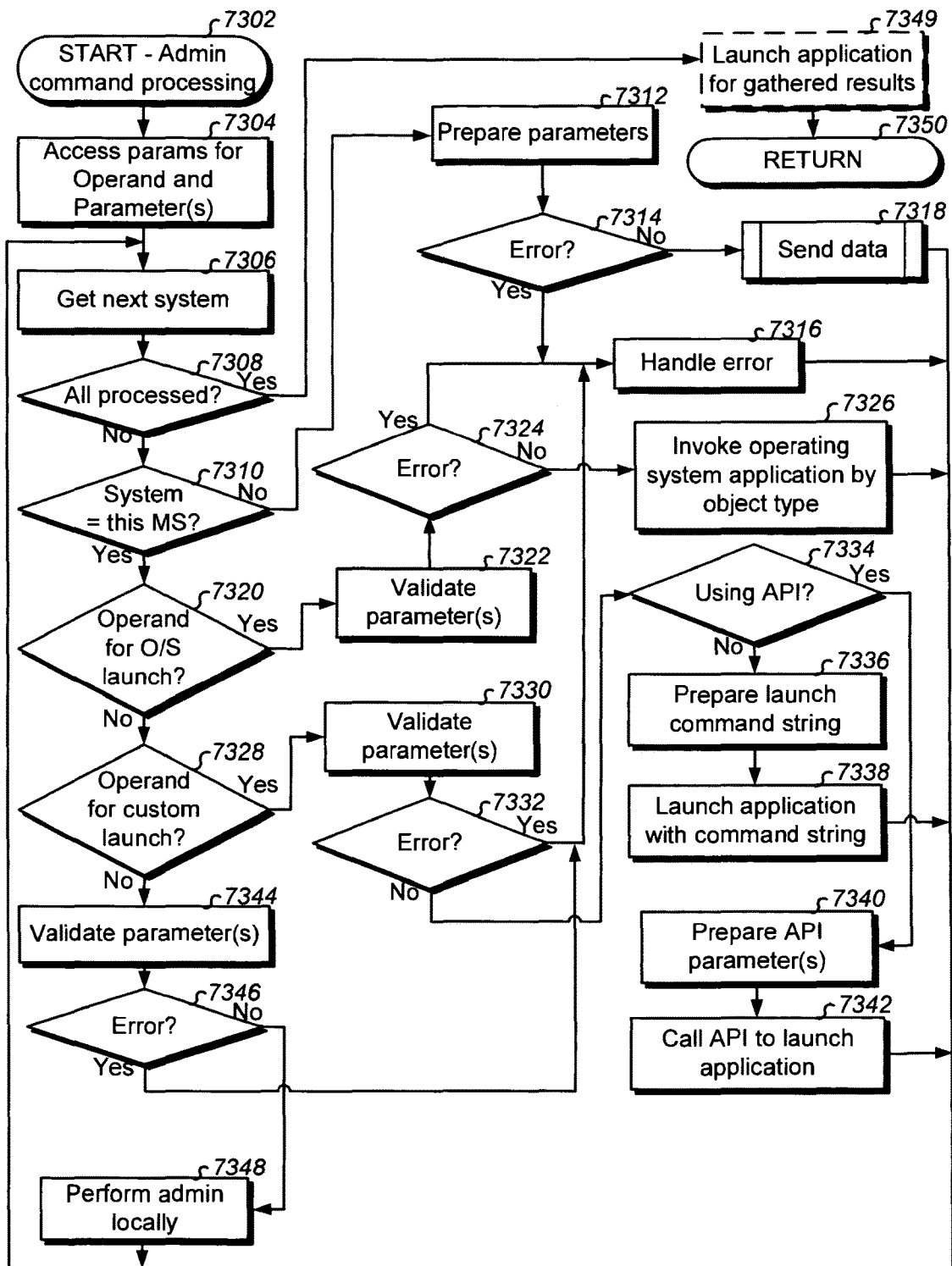


Fig. 73A

<i>Operand</i> ↓	PM	Preferred embodiment Administrative processing
201	C	<p>Administering an auto-dial # launches a MS phone number log interface with the auto-dial # parameter for searching. Preferably, both the outgoing and incoming logs are searched. The auto-dial # parameter can be a wildcard (pattern) for matching. In one embodiment, all occurrences found in history are presented with their date/time stamps, log found in, and perhaps other information, of the call and when it took place. In another embodiment, the most recent occurrence from a particular log is presented, and perhaps in an interface which enables calling the # with a minimal user action. The search takes place as though the user manually launched the search, entered the auto-dial # for the search, and then was presented with the result(s) in an appropriate administration interface. Appropriate MS storage is updated and subsequently processed as though the user manually performed the search. Preferably, the administration cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Subsequent administration includes modifying the log entry(s) as desired (e.g. add a 1 prefix since caller id may not have maintained one when it is needed for auto-dial). A new parameter can be specified for which log(s) to search.</p>
203	S	<p>Administering a weblink launches a search to MS browser history with the weblink parameter (and with the params parameter if specified) for searching. The weblink parameter can be a wildcard (pattern) for matching. In one embodiment, all occurrences found in history are presented with their date/time stamps, folder found in, and perhaps other information, of the link and when it was invoked. In another embodiment, the most recent occurrence from a particular invocation is presented, and perhaps in an interface which enables invoking (transposing to) the weblink with a minimal user action. The search takes place as though the user manually launched the search, entered the weblink for the search, and then was presented with the result(s) in an appropriate administration interface. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the administration command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Subsequent administration includes modifying the weblink(s) as desired (e.g. change description).</p>

Fig. 73B-1

Operand ↓	PM	<u>Preferred embodiment Administrative processing</u>
205	C	<p>Administrating an email causes searching a MS email system with search criteria of the email parameter string. The email parameter string can specify searching any email fields for any values including wildcarding (patterns for matching). Each field is referenced with a predefined name and then associated with a search criteria. For example, the email string of "subj:'personnel'; recip:'george@alltell.com'; body:'reduction in force'" causes searching all emails with a subject containing "personnel" and was sent to "george@alltell.com" and has a message body containing the string "reduction in force". To search for certain email containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:sent,inbox,company;" indicates to only search the email folders of sent, inbox, and company (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of email. Wildcarding (pattern matching) is preferably inherent by searching for substrings. All occurrences found in history are presented with at least their date/time stamps, subject line, sender and recipient, and perhaps other information, of the email and when it took place. In another embodiment, the most recent occurrence from searched folders is presented, and perhaps in an interface which enables appropriate MS email system processing from that point forward (e.g. when processed at local MS). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s) in an appropriate administration interface. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the administration command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Subsequent administration includes modifying the email(s) as desired, and perhaps having an option to send/resend.</p>

Fig. 73B-2

<i>Operand</i> ↓	PM	<u>Preferred embodiment Administrative processing</u>
207	C	<p>Administrating an sms message causes searching a MS sms messaging system with search criteria of the sms message parameter string. The message parameter string can specify searching any message fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria. For example, the sms message string of "recip:'2144034071@nextel.com,9725397137@lboxsrv.com';" causes searching all messages to the sought recipients. To search for certain messaging containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:outgoing" indicates to only search the outgoing folder (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of messages. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In the preferred embodiment, all occurrences found in history are presented with at least their date/time stamps, message, sender and recipient, and perhaps other information, of the message and when it took place. In another embodiment, the most recent occurrence from searched folders is presented, and perhaps in an interface which enables appropriate MS messaging system processing from that point forward (e.g. when processed at local MS). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s) in an appropriate administration interface. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the administration command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Subsequent administration includes modifying the sms message(s) as desired, and perhaps having an option to send/resend.</p>

Fig. 73B-3

Operand ↓	<u>PM</u>	<u>Preferred embodiment Administrative processing</u>
209	C	<p>Administrating a broadcast email causes searching a MS email system with search criteria of the email parameter string. The email parameter string can specify searching any email fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria. For example, the email string of "subj:'personnel';recip:'george@alltell.com';body:'reduction in force'" causes searching all emails with a subject containing "personnel" and was sent to "george@alltell.com" and has a message body containing the string "reduction in force". To search for certain email containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:sent,inbox,company;" indicates to only search the email folders of sent, inbox, and company (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of email. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In the preferred embodiment, all occurrences found in history are presented with at least their date/time stamps, subject line, sender and recipient, and perhaps other information, of the email and when it took place. In another embodiment, the most recent occurrence from searched folders is presented, and perhaps in an interface which enables appropriate MS email system processing from that point forward (e.g. when processed at local MS). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s) in an appropriate administration interface. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the administration command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Subsequent administration includes modifying the email(s) as desired, and perhaps having an option to send/resend.</p>

Fig. 73B-4

<i>Operand</i> ↓	PM	<u>Preferred embodiment Administrative processing</u>
211	C	Administrating a broadcast sms message causes searching a MS sms messaging system with search criteria of the sms message parameter string. The message parameter string can specify searching any message fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria. For example, the sms message string of "recip:'2144034071@nextel.com,9725397137@lbxsrv.com';" causes searching all messages to the sought recipients. To search for certain messaging containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:outgoing" indicates to only search the outgoing folder (no specification preferably indicates to search all folders)). Those skilled in the art recognize many useful syntaxes for searching any characteristics of messages. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In the preferred embodiment, all occurrences found in history are presented with at least their date/time stamps, message, sender and recipient, and perhaps other information, of the message and when it took place. In another embodiment, the most recent occurrence from searched folders is presented, and perhaps in an interface which enables appropriate MS messaging system processing from that point forward (e.g. when processed at local MS). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s) in an appropriate administration interface. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the administration command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Subsequent administration includes modifying the message(s) as desired, and perhaps having an option to send/resend.
213	O	See Find Command for identical processing this LBX release.

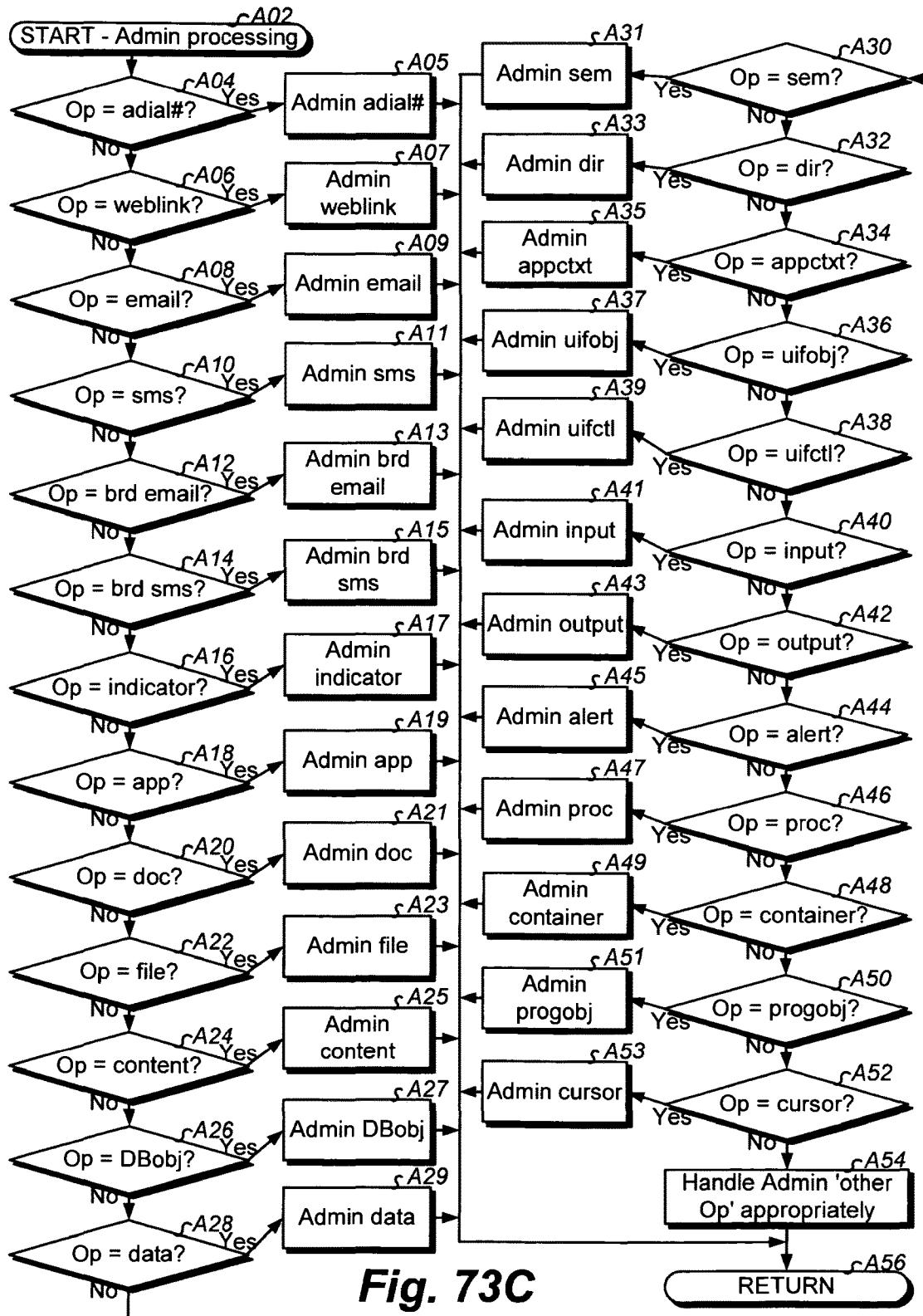
Fig. 73B-5

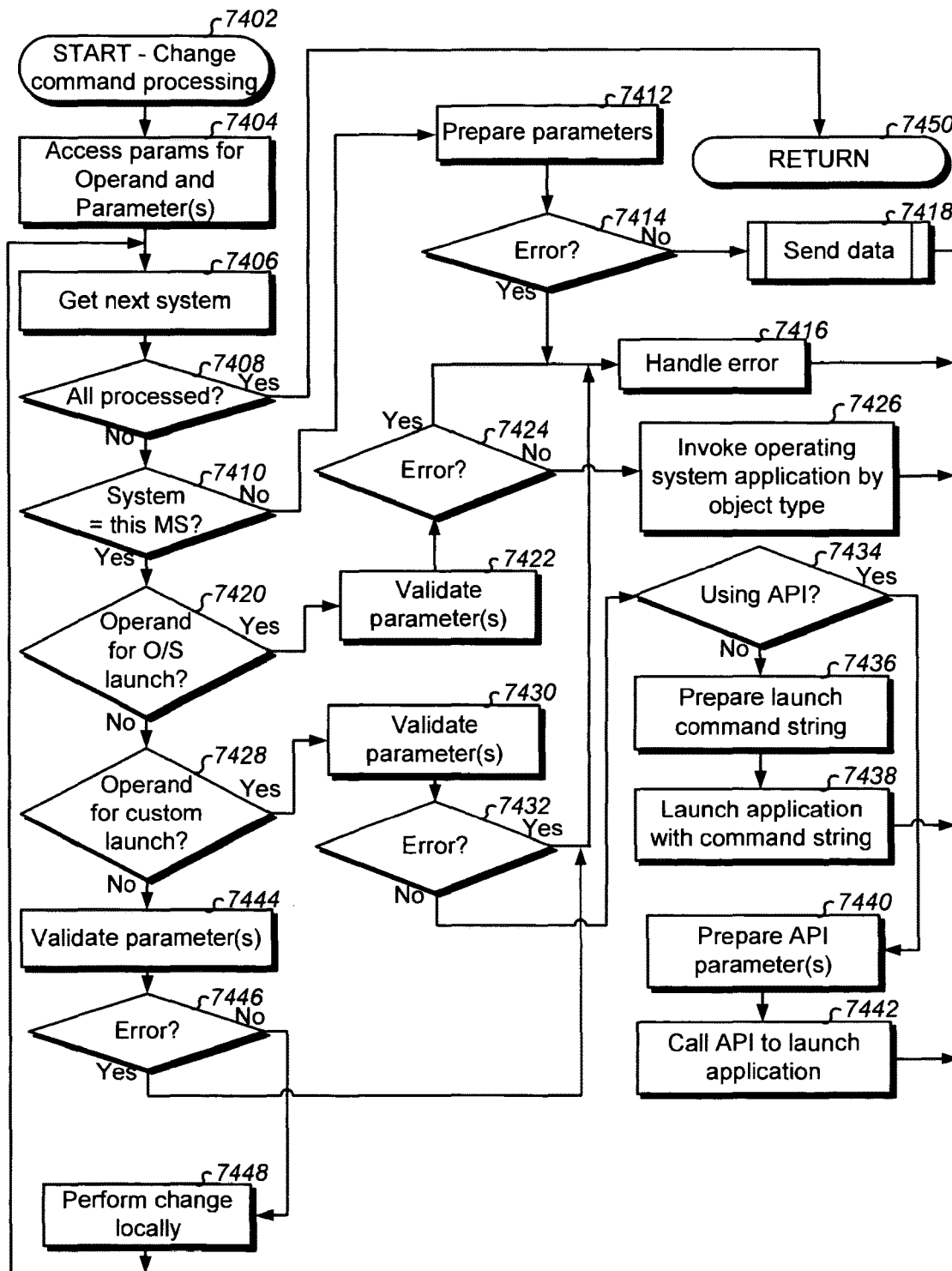
<i>Operand</i>	PM	<u>Preferred embodiment Administrative processing</u>
215	C	Administrating an application causes searching the MS for application (and with the params parameter if specified). The app parameter is preferably an executable name. Providing a more defined partial or full path to the application parameter will validate that it is found there. The app parameter string preferably supports wildcarding. In the preferred embodiment, all occurrences found on the MS and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information. In another embodiment, all parts which are linked to the executable are identified with their paths, date/time stamps, size, and perhaps attributes when a symbol file is specified with a new parameter. The symbol file is output from a link process and can be used to identify all executable parts such as dynamic link libraries, linked binaries, and any other executable binary file involved with the application. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s) in an appropriate administration interface (e.g. a properties edit user interface). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the administration command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. Subsequent administration includes modifying the configuration, startup parameters, or any other environmental variables of the application.
217	S	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
219	S	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
221	O	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
223	C	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
225	O	See Invoke Command for identical processing this LBX release.
227	O	See Invoke Command for identical processing this LBX release.
229	S	See Invoke Command for identical processing this LBX release.
231	C	See Invoke Command for identical processing this LBX release.
233	S	See Invoke Command for identical processing this LBX release.
235	O	See Invoke Command for identical processing this LBX release.
237	O	See Invoke Command for identical processing this LBX release.
239	O	See Invoke Command for identical processing this LBX release.

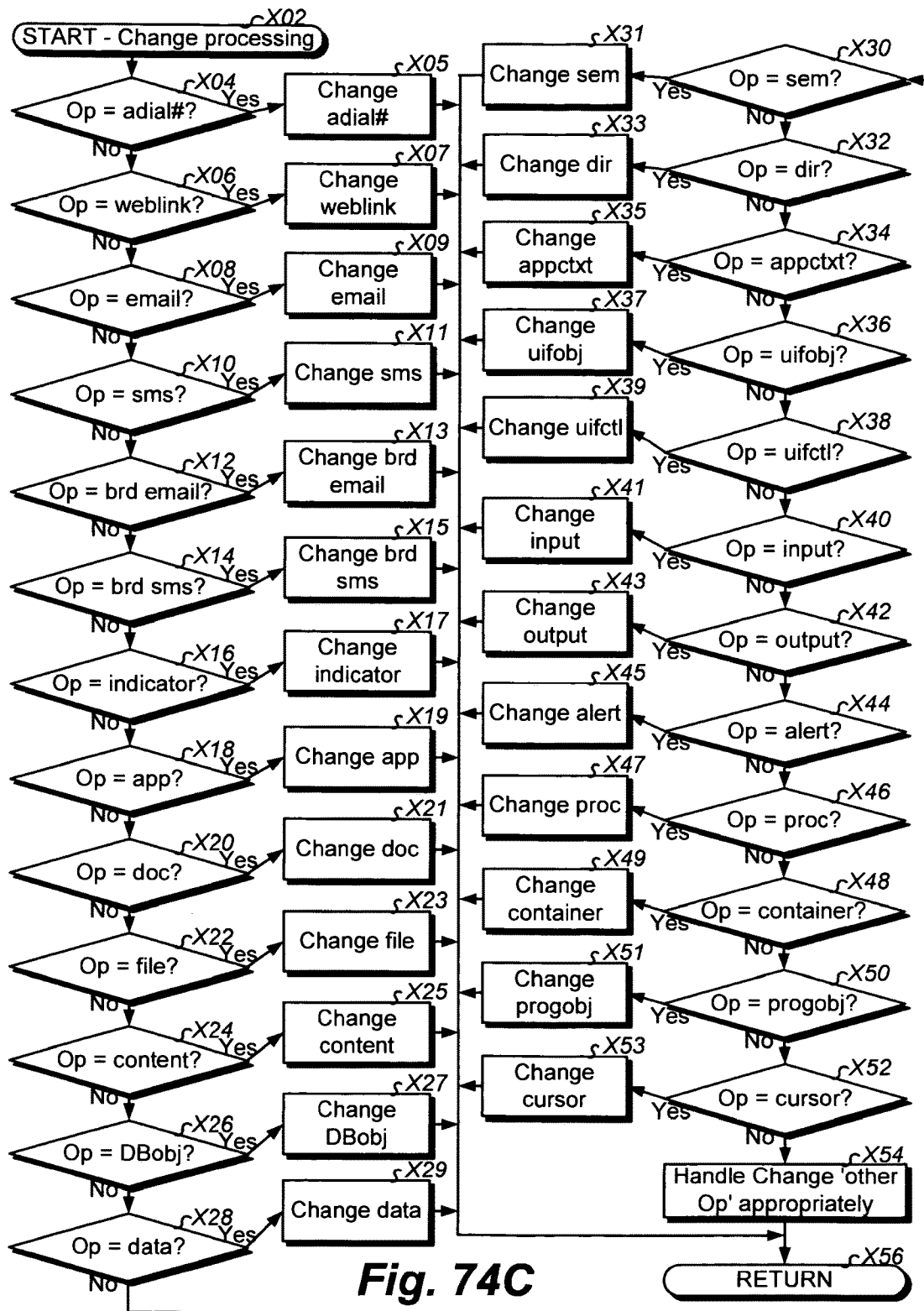
Fig. 73B-6

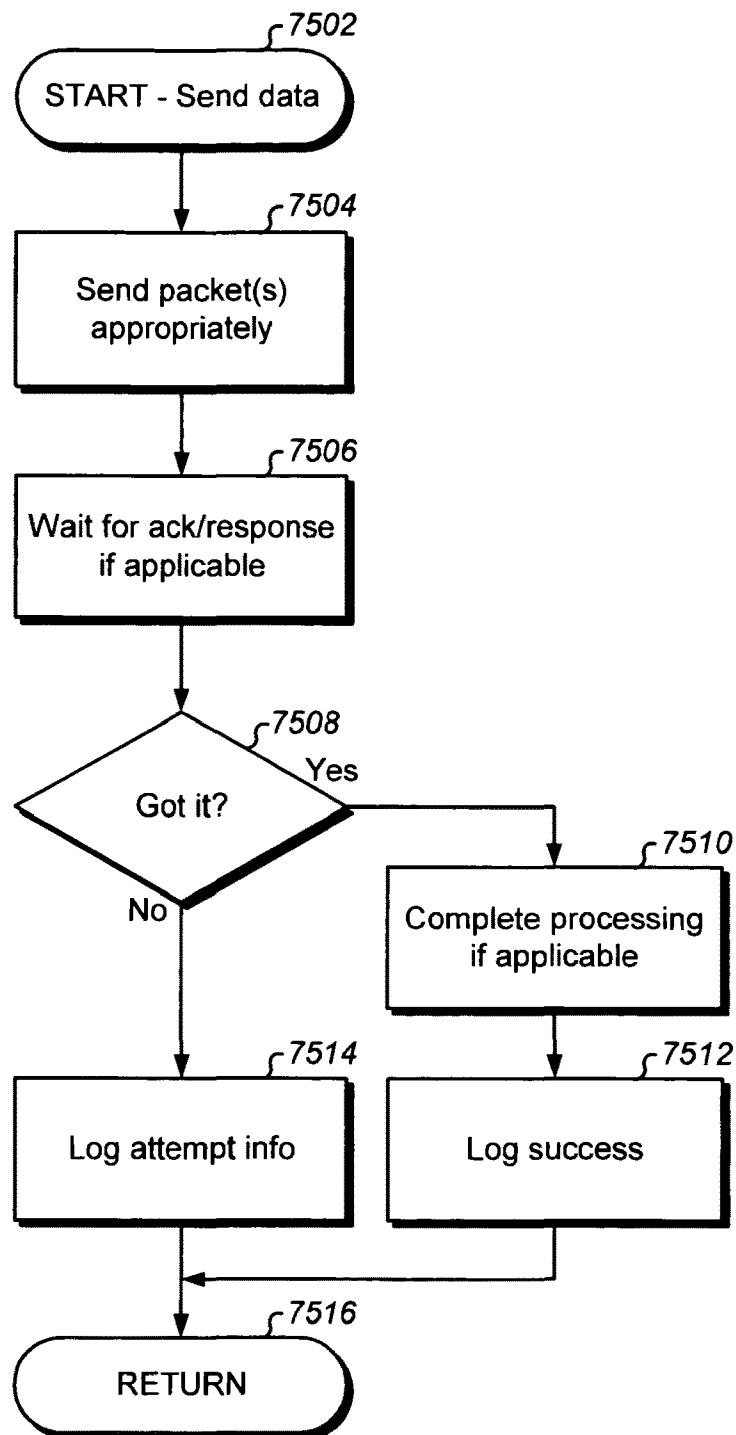
<i>Operand</i> ↓	PM	<u>Preferred embodiment Administrative processing</u>
241	S	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
243	O	See Invoke Command for identical processing this LBX release.
245	S	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
247	O	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
249	O	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
251	C	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
253	C	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
...		

Fig. 73B-7



**Fig. 74A**



**Fig. 75A**

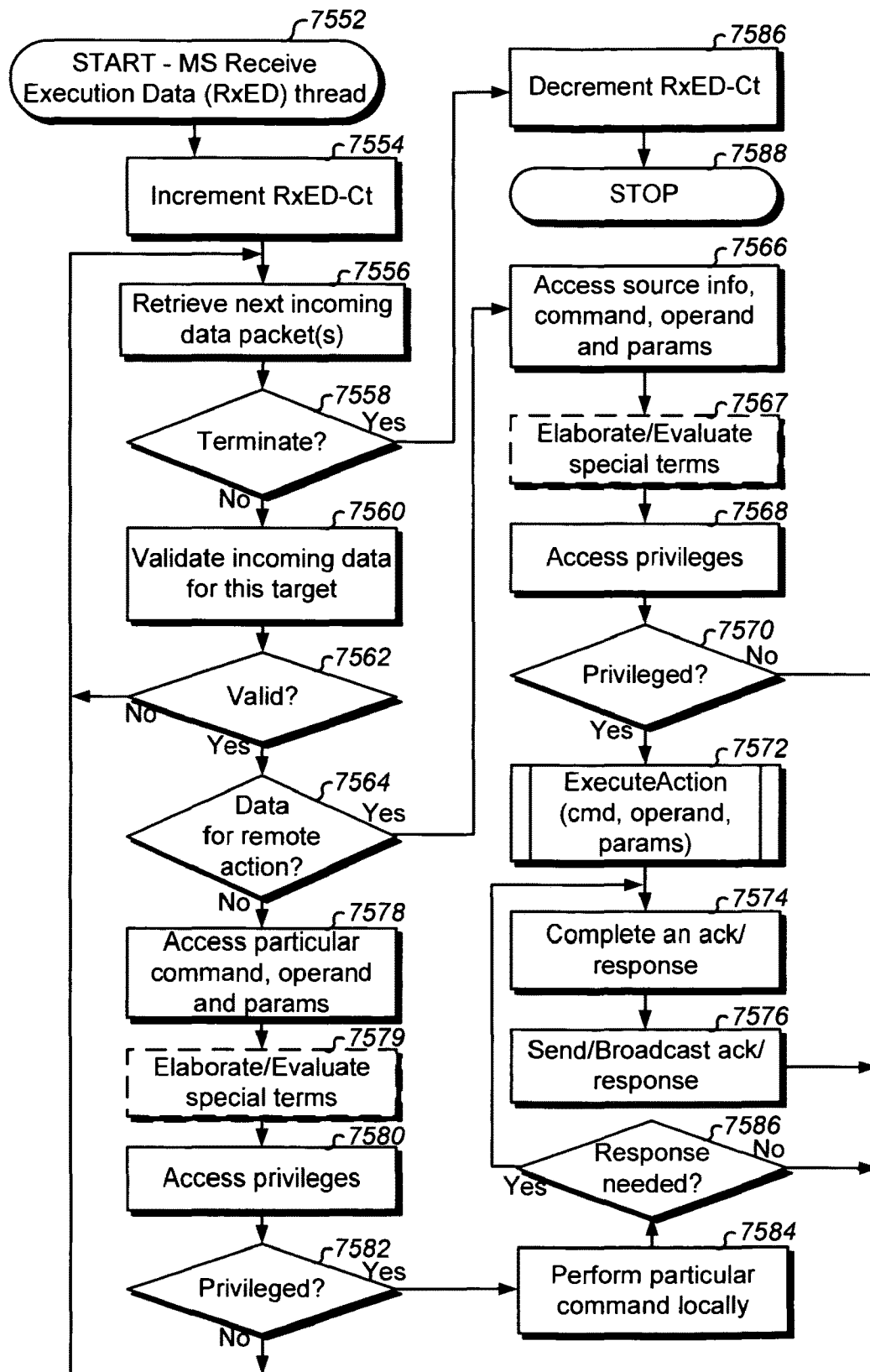
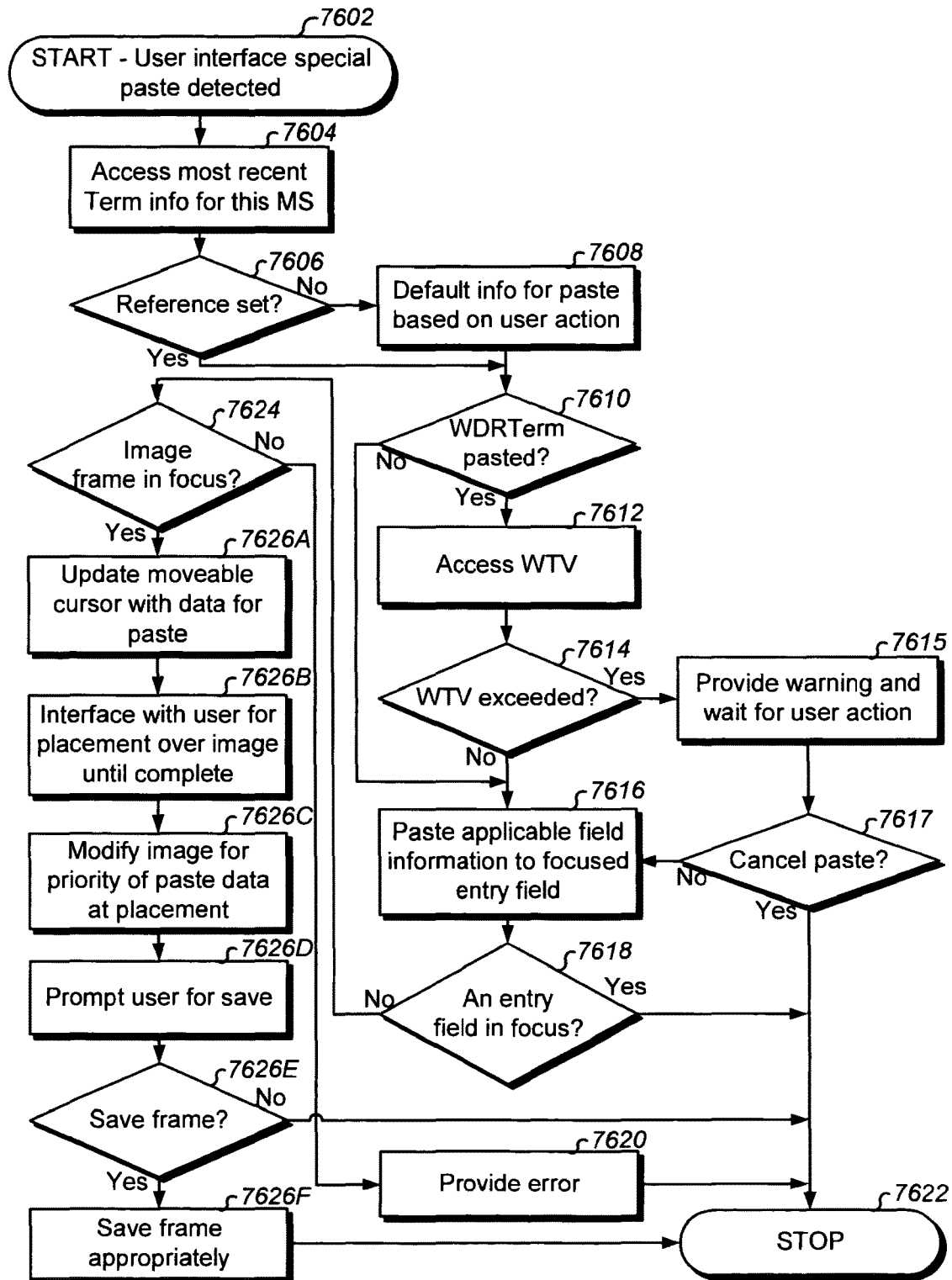
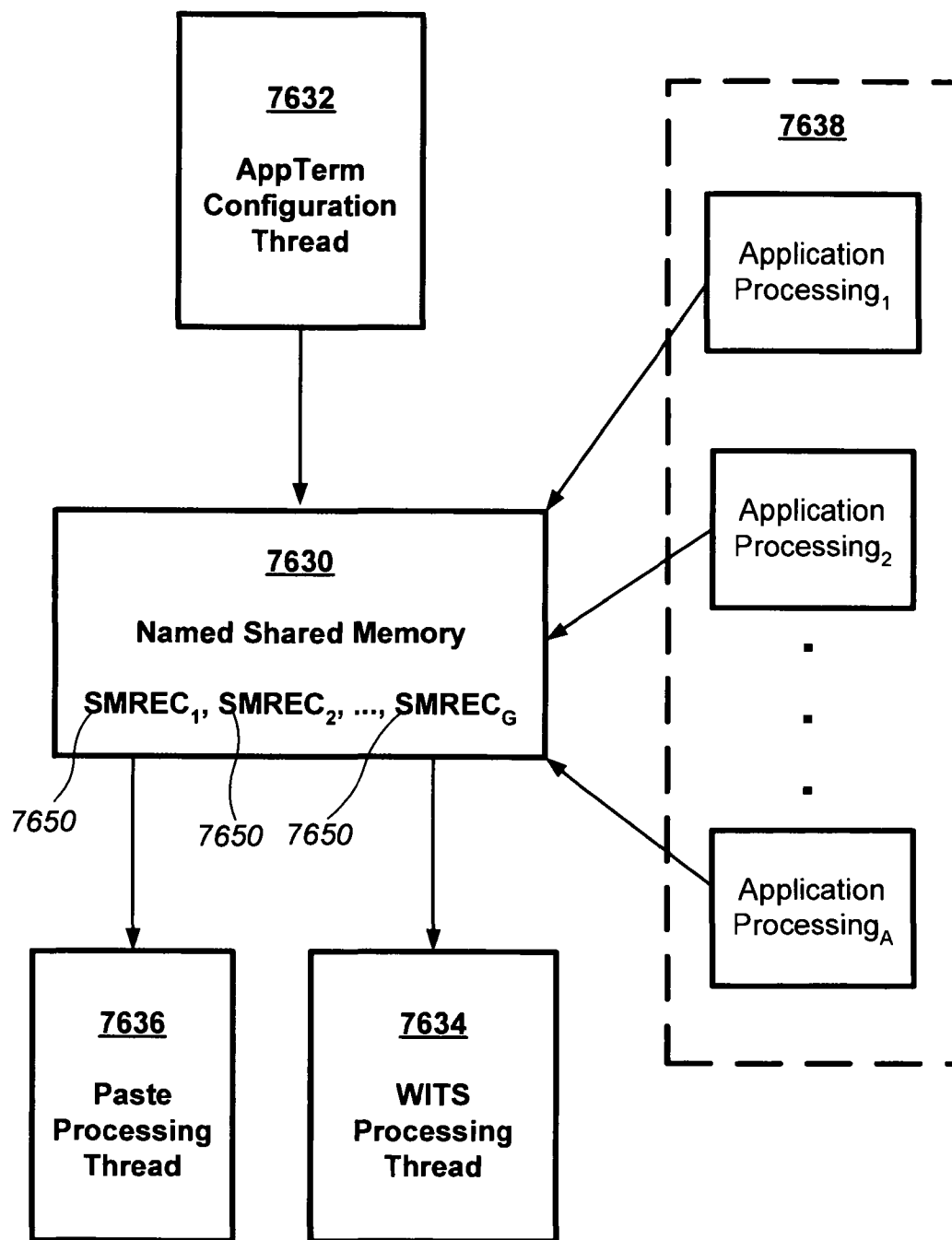
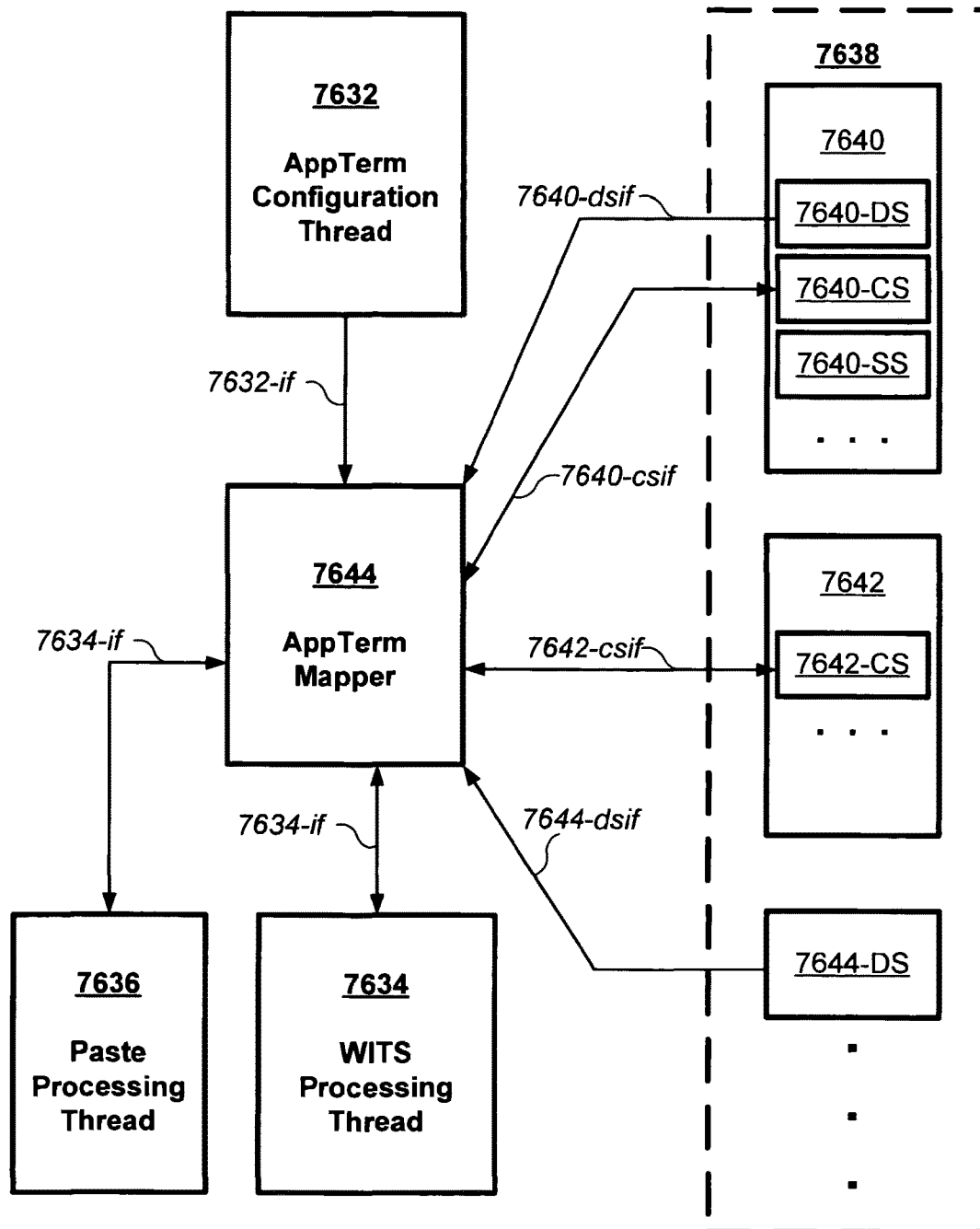
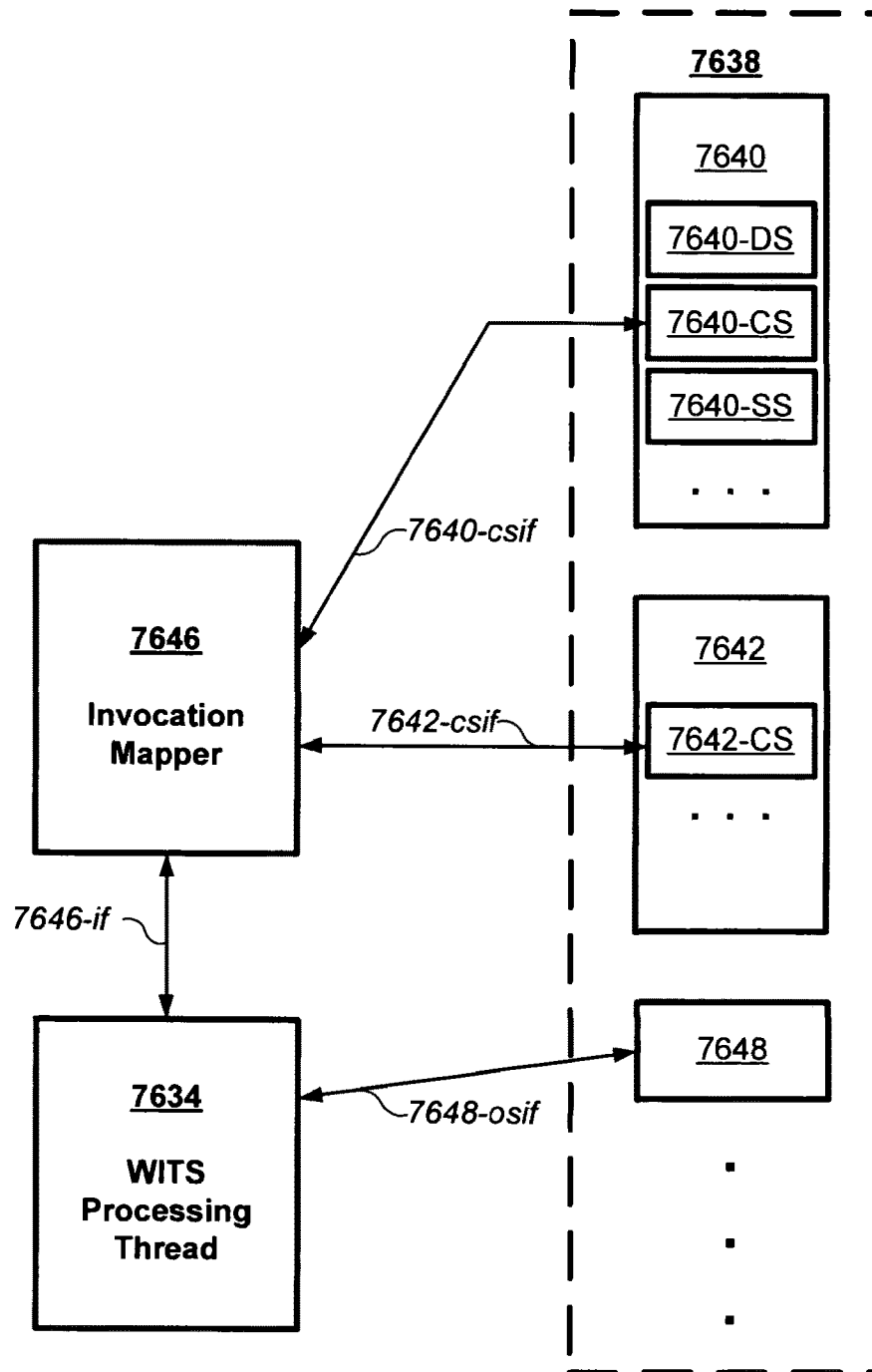


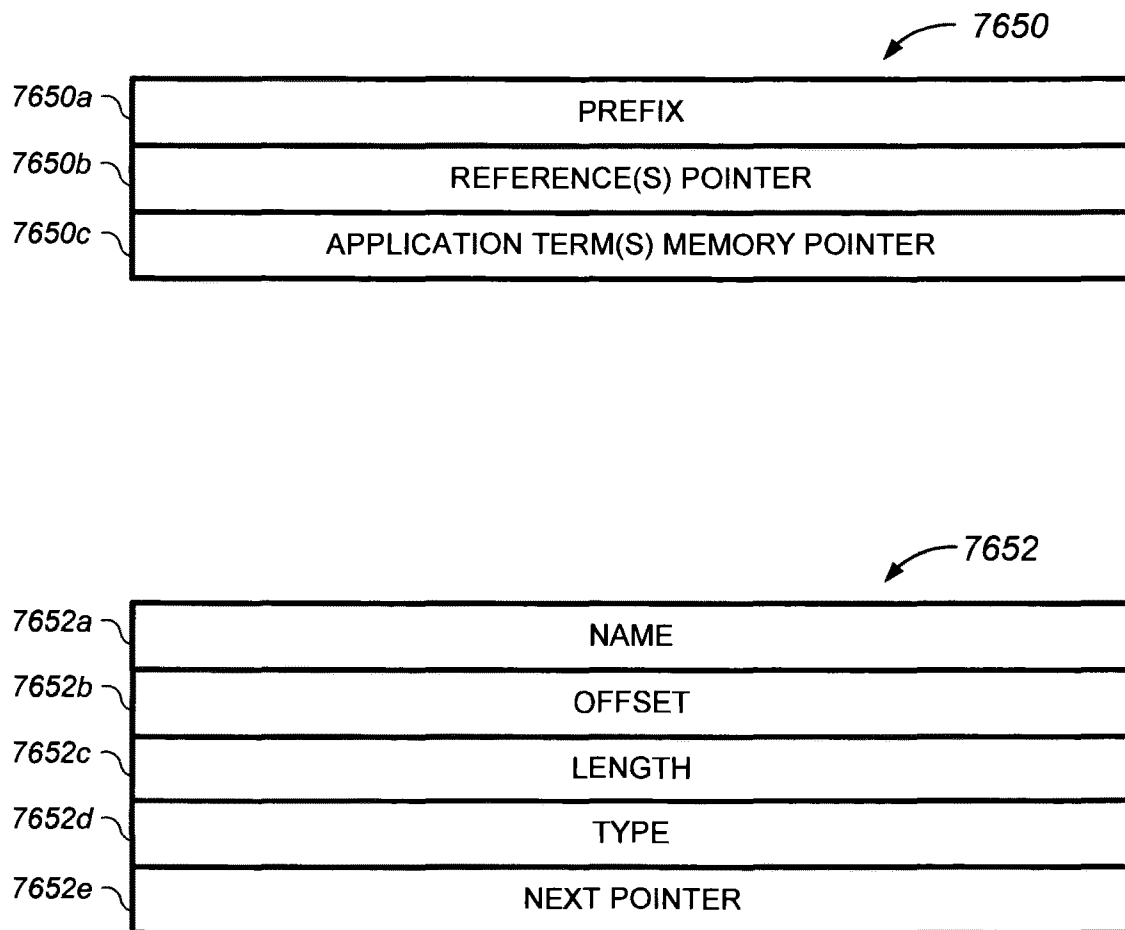
Fig. 75B

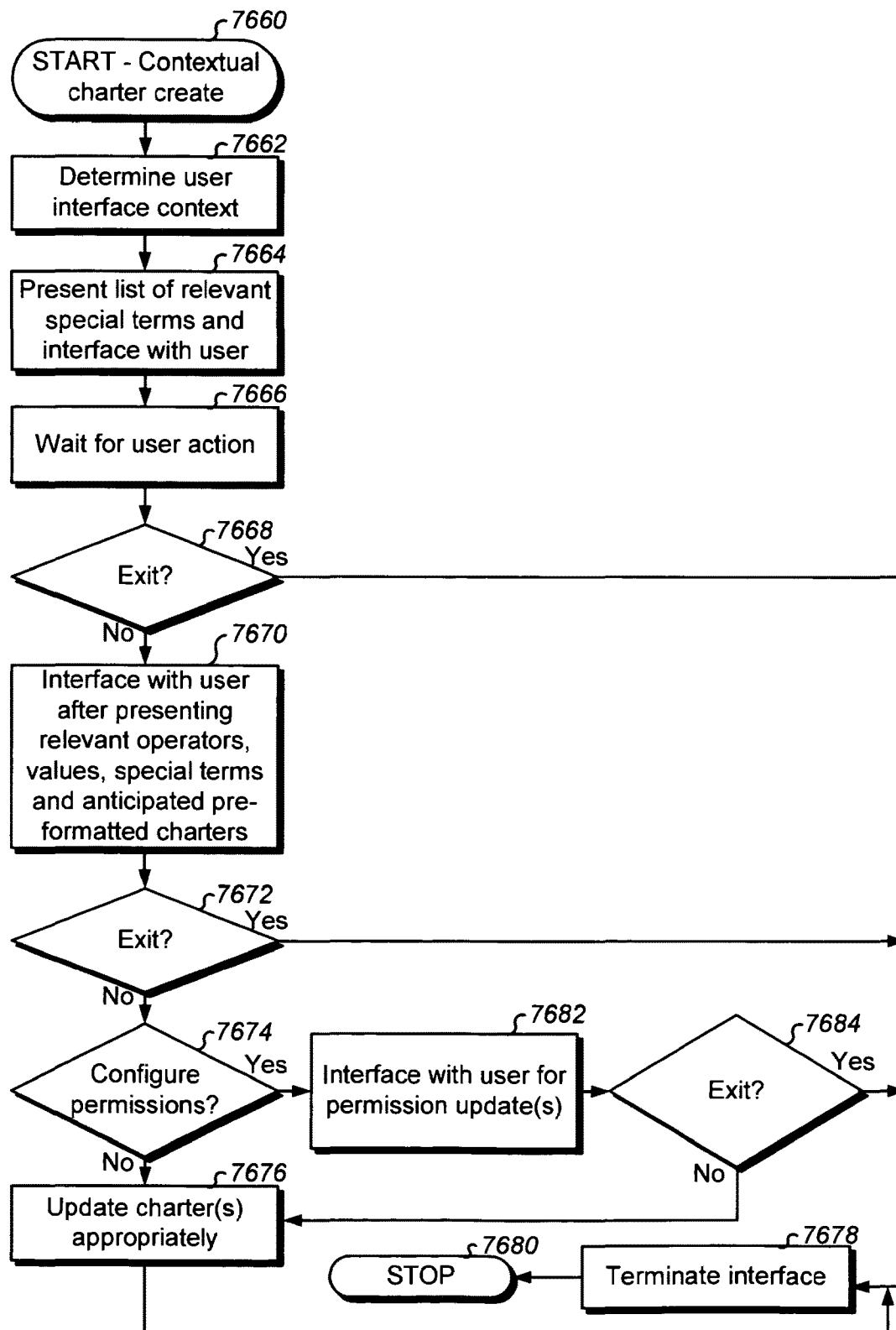
**Fig. 76A**

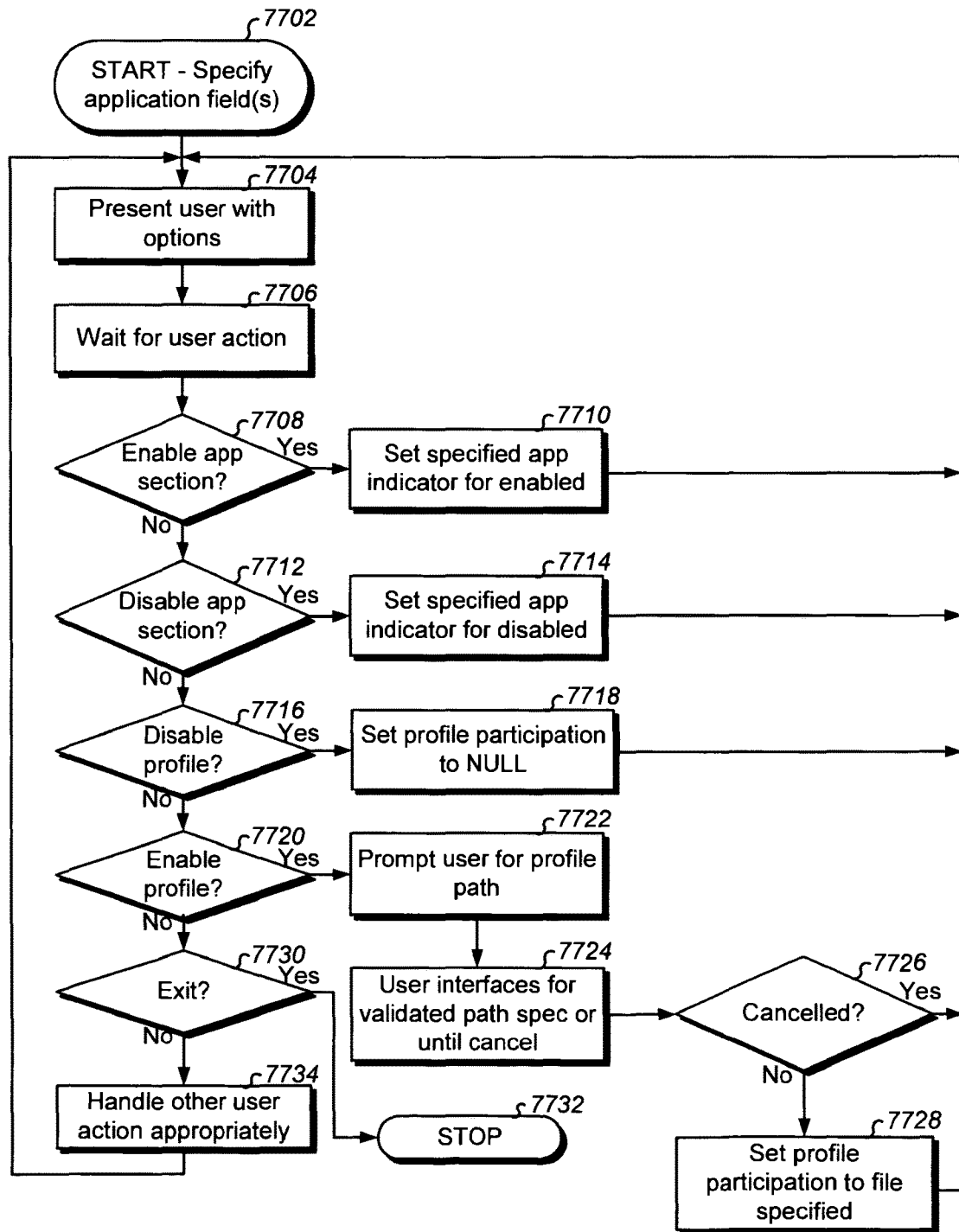
**Fig. 76B-1**

**Fig. 76B-2**

**Fig. 76B-3**

**Fig. 76C**

**Fig. 76D**

**Fig. 77**

U.S. Patent

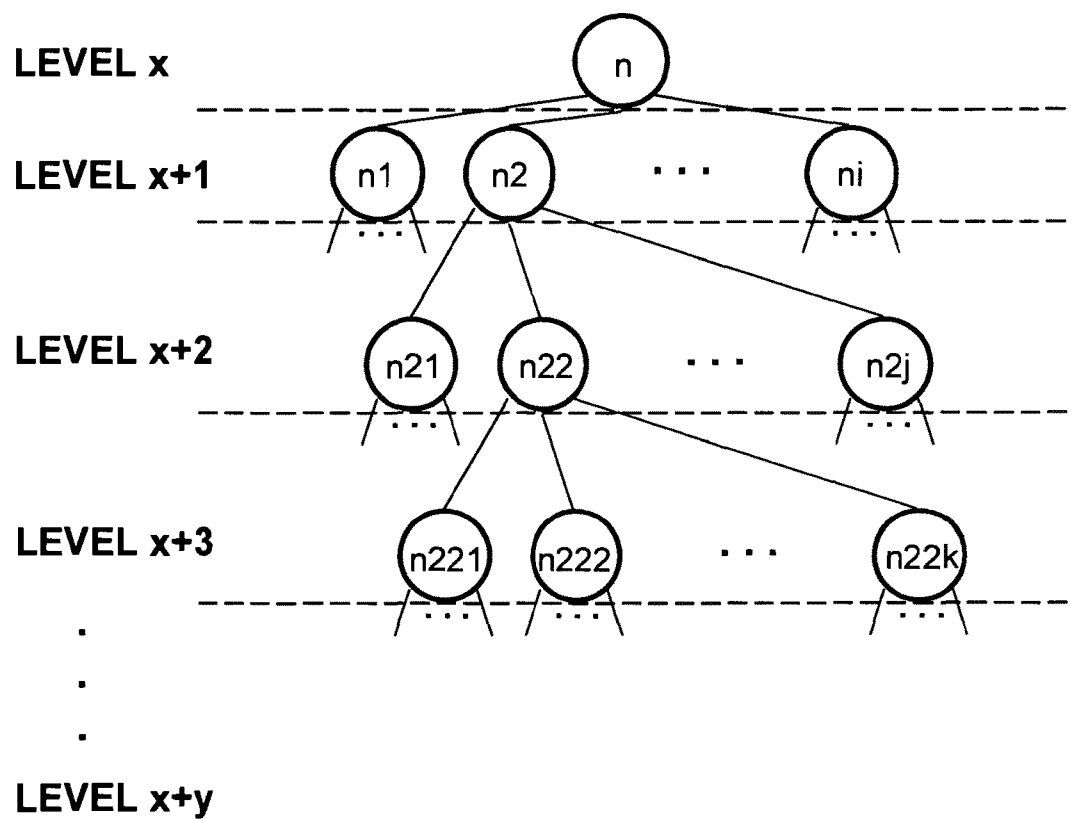
Nov. 19, 2019

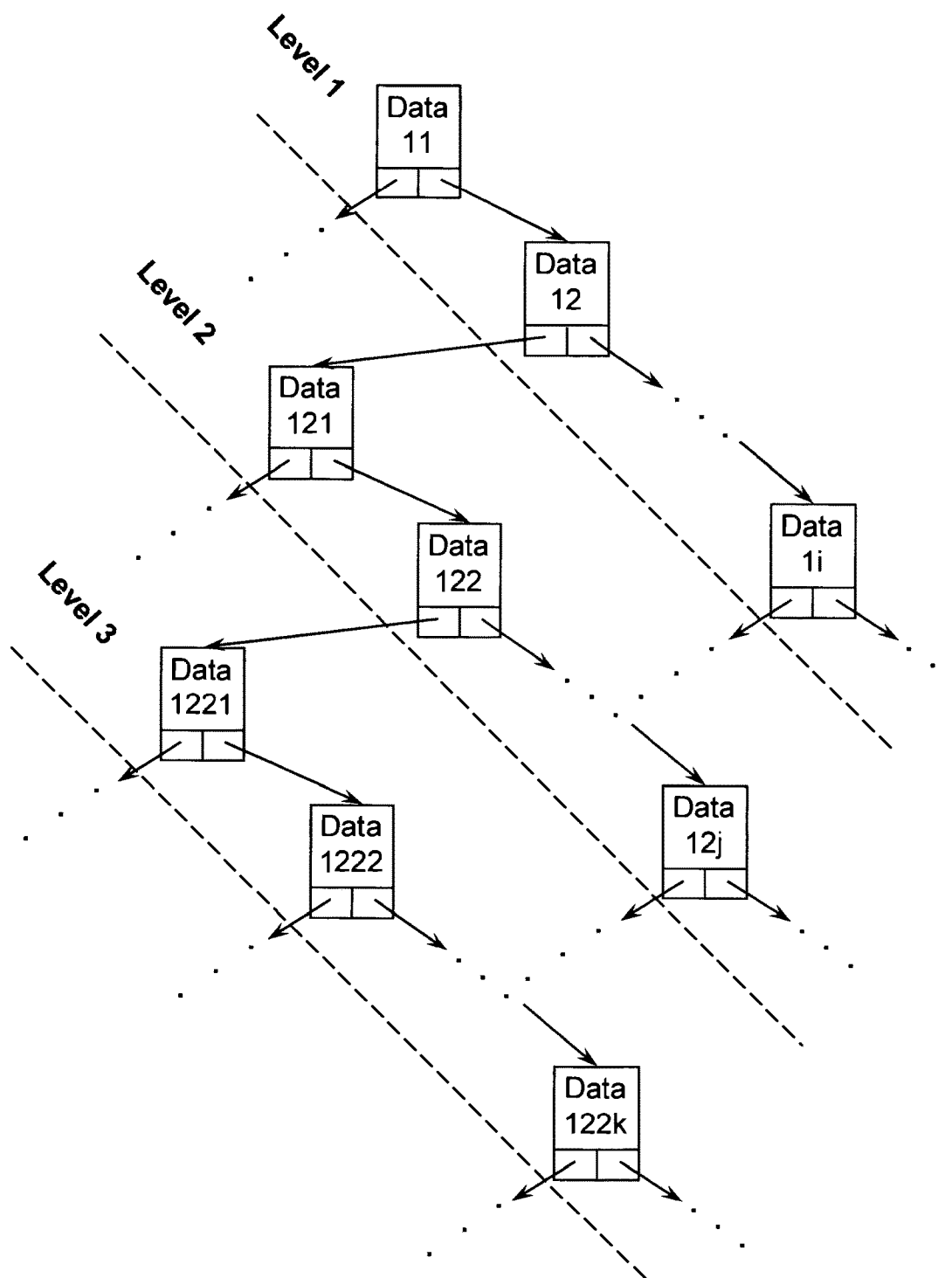
Sheet 269 of 322

US 10,477,994 B2

```
...
<home>
  ...
  <city>Moorestown</city>
  <state>New Jersey</state>
  ...
</home>
...
<interests >
basketball;programming; running; football
</interests>
...
<hangouts>
  ...
  <morning>Starbucks</morning>
  <lunch>Jammin's;Mongolian Barbeque</lunch>
  <evening>Confettis;Jimbos</evening>
  ...
</hangouts>
...
```

Fig. 78

**Fig. 79A**

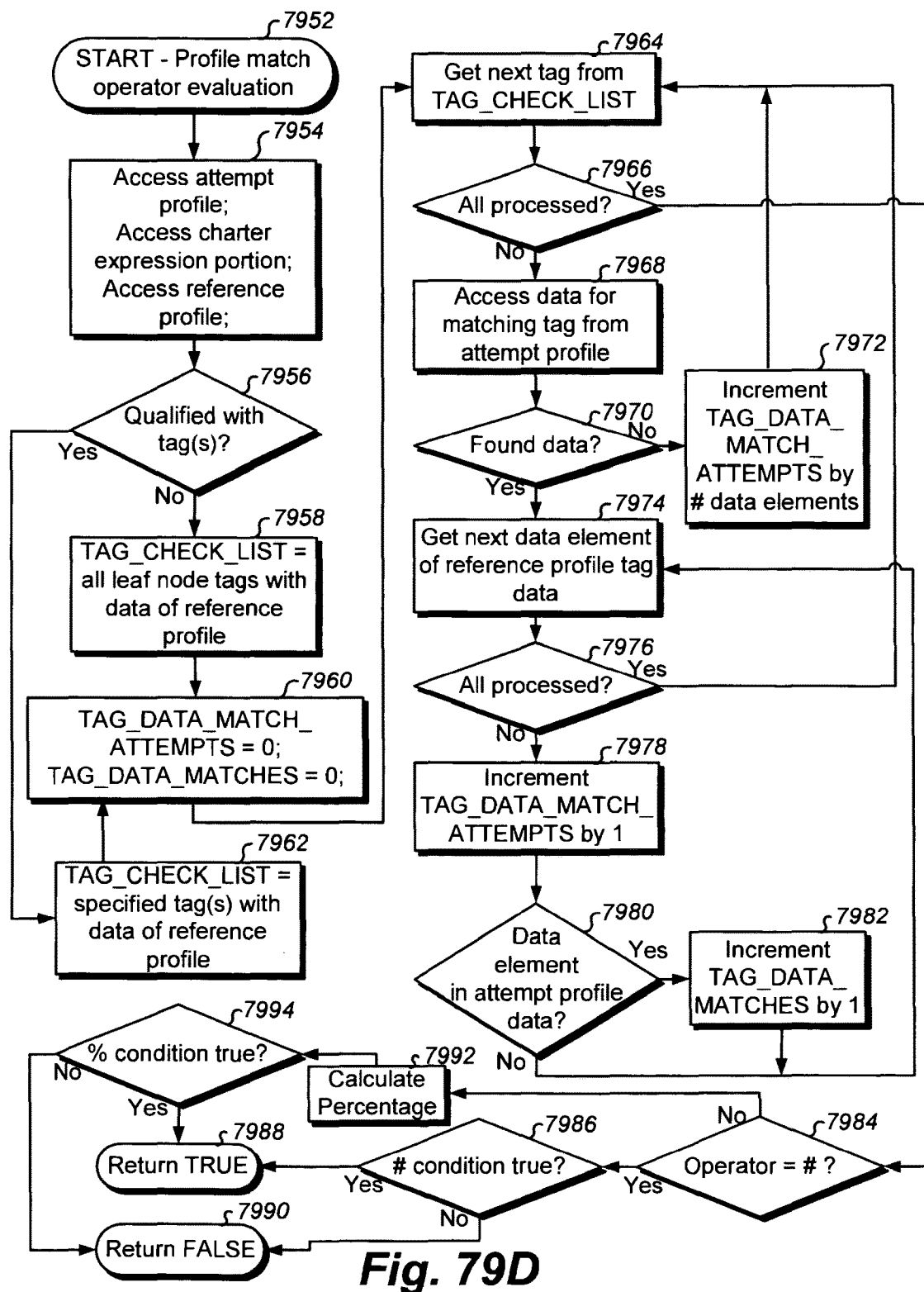
**Fig. 79B**

```

.
.
.
typedef struct xml_node {
    struct xml_node *descend1st; // Points to first descendant tag in XML doc
    struct xml_node *peer_down; // Points to same level tag down XML doc
    struct xml_node *ascendant; // Points to ascending tag in XML doc
    struct xml_node *peer_up; // Points to same level tag up XML doc
    char data_type; // Type of data @ data pointer
    unsigned char *data; // Typecast-able pointer to data
} XML_NODE;
.
.
.
XML_NODE *Lprofile; // Root node to XML doc tag tree for Local profile
.
.
.
XML_NODE *Rprofile; // Root node to XML doc tag tree for Remote/Received profile
.
.
.

```

Fig. 79C



U.S. Patent

Nov. 19, 2019

Sheet 274 of 322

US 10,477,994 B2

8000
↙

	<u>appname</u>	<u>Application Description</u>	<u>Status</u>
8002a	source	Configurable MS ID	Registered
8002b	profile	% and # operator object	Registered
8002c	email	Electronic mail	Registered
8002d	calendar	Electronic calendar	Registered
8002e	ab	Electronic address book	Registered
8002f	phone	Electronic phone	Registered
8002g	emergency	Emergency use	Registered
8002h	loc	LBX locational data sharing	Registered
8002i	rfid	Radio Frequency Identification	Registered
8002j	hotspot	Wifi/Wimax/Xan	Registered
8002k	services	Published services for service propagation	Registered
8002l	statistics	MS statistics (may be shared between MSs)	Registered
8004a	traffic	Traffic Reports	RFP
8004b	appliance	Appliance Control	RFP
8004c	acctmgt	Account Management (ATM, Banking)	RFP
8004d	transport	Public Transportation (Bus, Taxi, Air, Train)	RFP
8004e	carpool	Automotive "car-pooling"	RFP
8004f	advertise	Advertising	RFP
8004g	news	News	RFP
8004h	media	Video, Pictures	RFP
8004i	parking	Parking lot awareness	RFP
8006a	employ	Employment, Job Awareness	Presented
8006b	real	Real Estate	Presented
8008a	personal	Personal Use	Tabled

Fig. 80A

↖ 8002a

Field 1100k reference	Description
appfld.source.id.X	MS ID context sensitive default value (X in [email, phone, calendar, ab, rfid, ip (uses first), one for each candidate id for contextual elaboration of an Expression...])
appfld.source.type	MS type
appfld.source.mfr	MS manufacturer
appfld.source.serno	MS serial number
appfld.source.ip	Current delimited (e.g. semicolon) IP address(es) of MS (may have > 1)
...	...

↖ 8002b

Field 1100k reference	Description
appfld.profile.contents	MS profile info (e.g. for % or # operator)
...	...

↖ 8002c

Field 1100k reference	Description
appfld.email.source	Primary MS email app sender ID
appfld.email.default.X	Email compose defaults wherein X in [attribute.Y such that Y in [cod, urgent, charcode, one for each settable attribute for email...], salutation, doctype, recips, encrypt, compress, one for each email default variable...]
appfld.email.type	Email app type/name
appfld.email.pending.X	Pending email in progress of being composed: X in [see .default.X fields above, cdt, content, one for each email variable...]
appfld.email.last.X.Y.Z	Last email sent (i.e. X = sent) or last email received (i.e. X = rcvd) to/from Y (Y in [ANY, {id} such that id specific source/destination (e.g. joe@yahoo.com)]; Z in [see .pending.X fields above])
...	...

Fig. 80B-1

U.S. Patent

Nov. 19, 2019

Sheet 276 of 322

US 10,477,994 B2

↖ 8002e

Field 1100k reference	Description
appfld.ab.id	Override for AB identifier.
appfld.ab.default.X	AB entry compose defaults wherein X in [attribute.Y such that Y in [marker, color, font, size, one for each settable attribute for AB entry...], background, one for each AB entry default variable...]
appfld.ab.type	AB app type/name
appfld.ab.pending.X	Pending AB entry in progress of being composed: X in [see .default.X fields above, cdt, content, group, one for each AB entry variable...]
appfld.ab.last.X.Y.Z	Last AB entry created locally (i.e. X = local) or last AB entry received (i.e. X = other) to/from Y (Y in [ANY, {id} such that id specific source/destination (e.g. MSID4F3EB2398)]; Z in [see .pending.X fields above]
...	...

↖ 8002d

Field 1100k reference	Description
appfld.calendar.id	Calendar id override.
appfld.calendar.next.X	MS calendar next entry data X
appfld.calendar.nextavail.X	MS calendar app next available free slots
appfld.calendar.default.X	Calendar entry compose defaults wherein X in [attribute.Y such that Y in [cod, urgent, color, one for each settable attribute for calendar entry...], recips, camp, one for each calendar entry default variable...]
appfld.calendar.sched.X	MS calendar schedule for period X (X in [curweek, curmonth, curyear, etc])
appfld.calendar.type	MS calendar app type/name
appfld.calendar.pending.X	Pending calendar entry in progress of being composed: X in [see .default.X fields above, cdt, content, date/time(s), recurring, one for each calendar entry variable...]
appfld.calendar.last.X.Y.Z	Last calendar entry created locally (i.e. X = local) or last calendar entry received (i.e. X = other) to/from Y (Y in [ANY, {id} such that id specific source/destination (e.g. joe@yahoo.com)]; Z in [see .pending.X fields above]
...	...

Fig. 80B-2

U.S. Patent

Nov. 19, 2019

Sheet 277 of 322

US 10,477,994 B2

8002f

<u>Field 1100k reference</u>	<u>Description</u>
appfid.phone.id	MS phone app primary caller id
appfid.phone.default.X	Phone call defaults wherein X in [volume, encrypt, compress, camp, one for each phone application default variable...]
appfid.phone.caller	MS phone app caller id override
appfid.phone.log.X	MS phone app log file X (X in [out, in, missed, one for each log...])
appfid.phone.record.X	MS phone app record boolean for calls made, calls received, or specific numbers (e.g. MS IDs)
appfid.phone.ogm	MS phone app OGM
appfid.phone.dt.X	Date/time stamp for X (X in [tx (last call made), rx (last call received), missed (last call missed)])
appfid.phone.type	MS phone app type/name
appfid.phone.fwd	MS phone app forwarding setting for prioritized list. A prioritized list automatically tries the next entry if there is no answer or a failed outbound connection.
appfid.phone.ring	Ring setting = ring tone selection reference OR audio file reference.
appfid.phone.vibe	Vibration setting = None OR reference for vibration type.
appfid.phone.droplocs	MS phone dropped locations
appfid.phone.macro.X	Automated macros for ARU interfaces
appfid.phone.pwd.X	MS phone passwords for allowing calls to complete and for variable processing by caller.
appfid.phone.msg.X	Phone messages (new, saved, etc)
appfid.phone.blackout	MS phone blackout conditions (expressions including application in use, date/time(s), current location(s), any MS detectable condition)
appfid.phone.pending.X	Pending phone call: X in [see .default.X fields above, cdt, data, one for each phone call app variable...]; data is only present for peer to peer MS phone calls (it carries the voice call).
appfid.phone.last.X.Y.Z	Last phone call made (i.e. X = out) or last call received (i.e. X = in) to/from Y (Y in [ANY, {id} such that id specific source/destination (e.g. MSID4F3EB2398)]; Z in [see .pending.X fields above, edt]
...	

Fig. 80B-3

8002g

<u>Field 1100k reference</u>	<u>Description</u>
appfld.emergency.type	Emergency type (Police, Fire, Amber, Help, Caution, etc)
appfld.emergency.cdt	Emergency create date/time stamp
appfld.emergency.duration	Emergency anticipated duration
appfld.emergency.content.X	Emergency content.X (X in [type, alert, prefmeth, one field for each alert content section...])
appfld.emergency.method.X	MS emergency notify method X (X in [attribute.Y, how, where])
appfld.emergency.last.X	Last emergency WDR data: X in [self, other].
...	...

8002h

<u>Field 1100k reference</u>	<u>Description</u>
appfld.loc.blackout	Blackout criteria
appfld.loc.mode	Current MS mode
appfld.loc.geofence.X	MS configured geofence data
appfld.loc.halo.X	MS configured interest perimeter
appfld.loc.mark.X	X = # of saved location marks Y
appfld.loc.dcdb.X	Location activated delivery content for MS ID X
appfld.loc.beacon.X	Beacon a peer MS at the peer MS; Sending MS controls what is seen by receiving MS
...	...

8002i

<u>Field 1100k reference</u>	<u>Description</u>
appfld.rfid.id	Defaults appfld.source.id.rfid.
appfld.rfid.passive.X	MS Passive RFI capability
appfld.rfid.active.X	MS Active RFI capability
appfld.rfid.listen.X	Active RFID listening channel directions
appfld.rfid.seek.X	Passive RFID polling channel directions
...	...

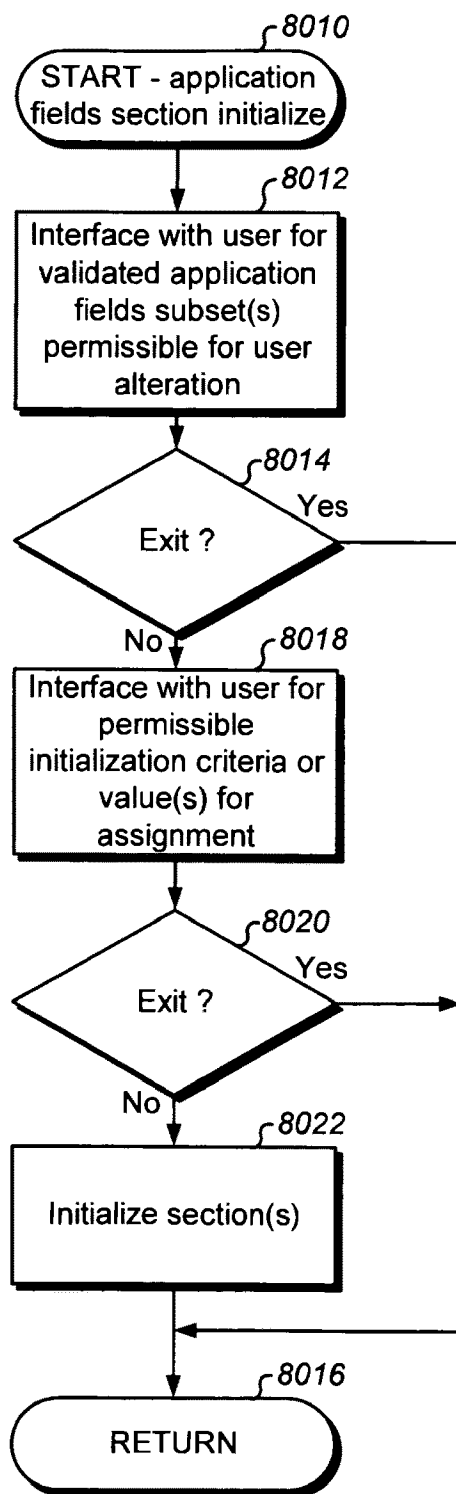
8002j

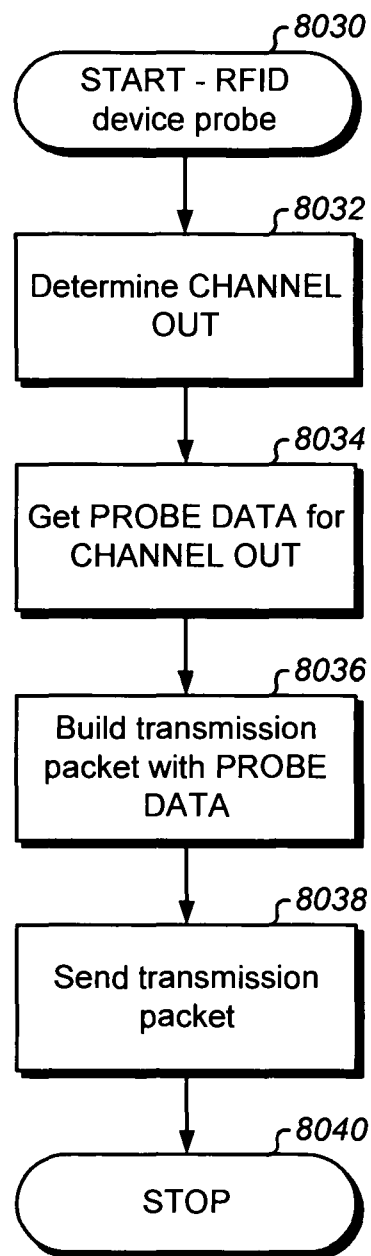
<u>Field 1100k reference</u>	<u>Description</u>
appfld.hotspot.listen	Listening boolean
appfld.hotspot.X	X number of Hotspot(s) (the information) automatically detected over time
...	...

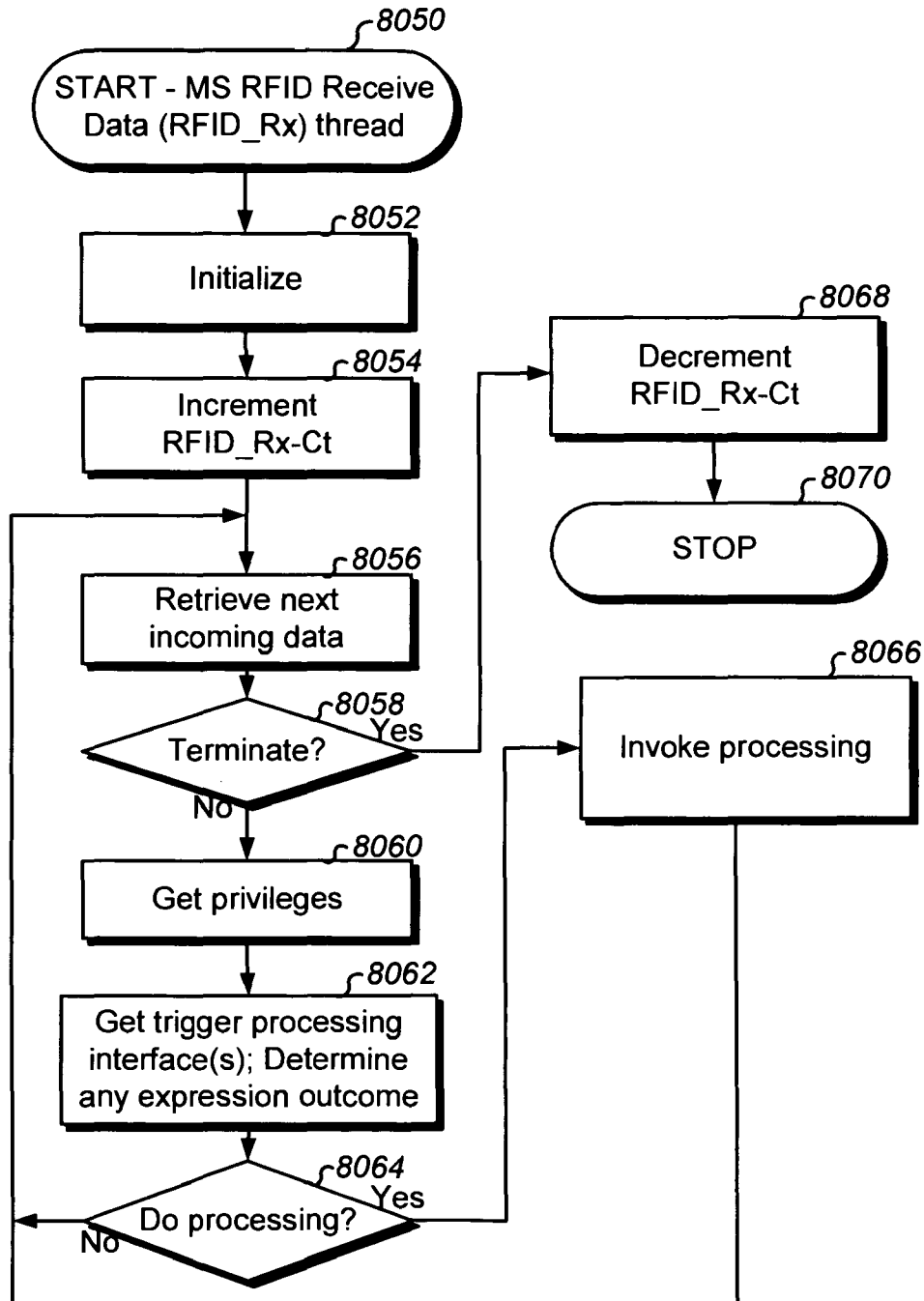
8002k

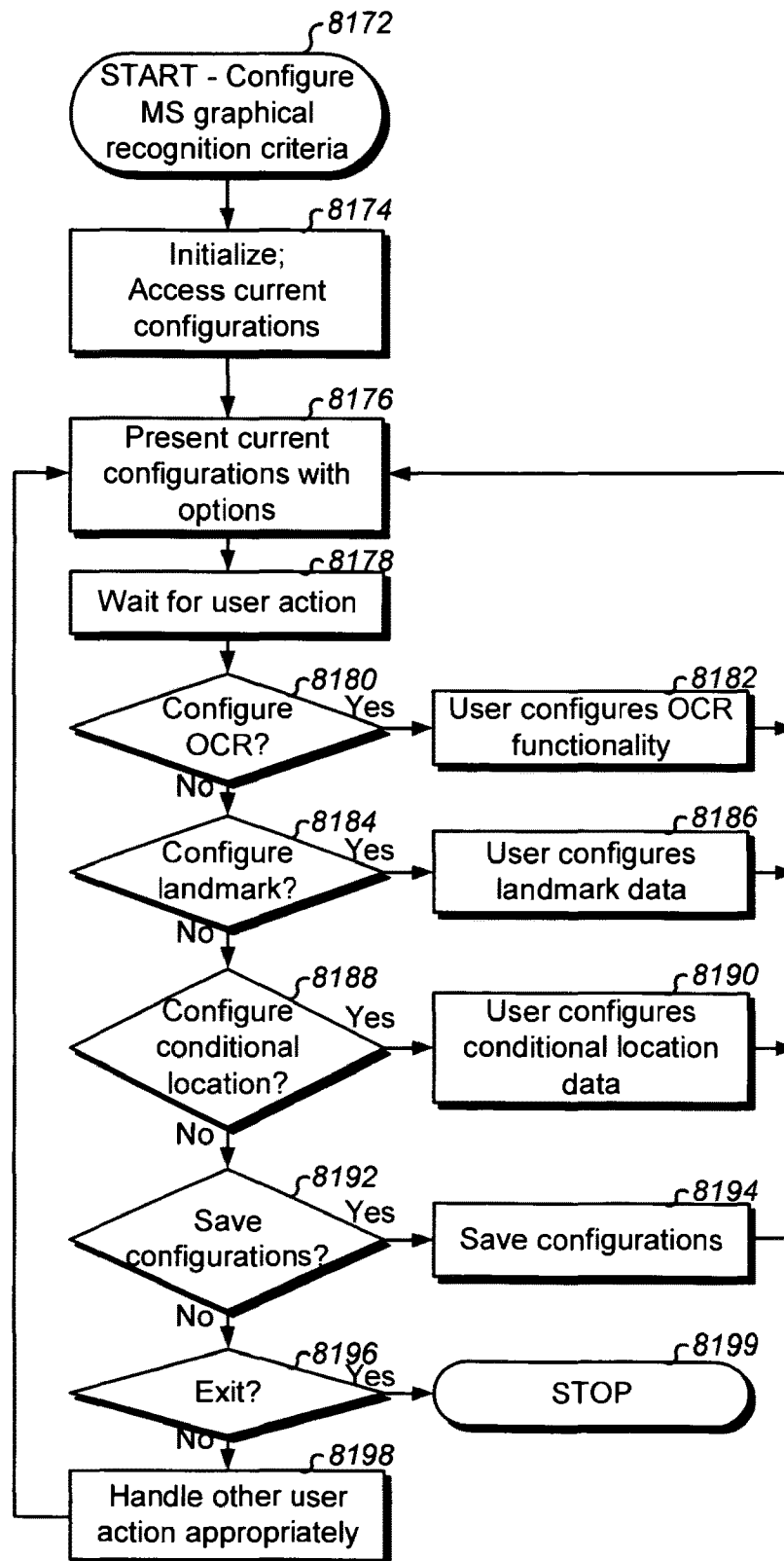
<u>Field 1100k reference</u>	<u>Description</u>
appfld.services.X	LN-expanse dynamic services information
...	...

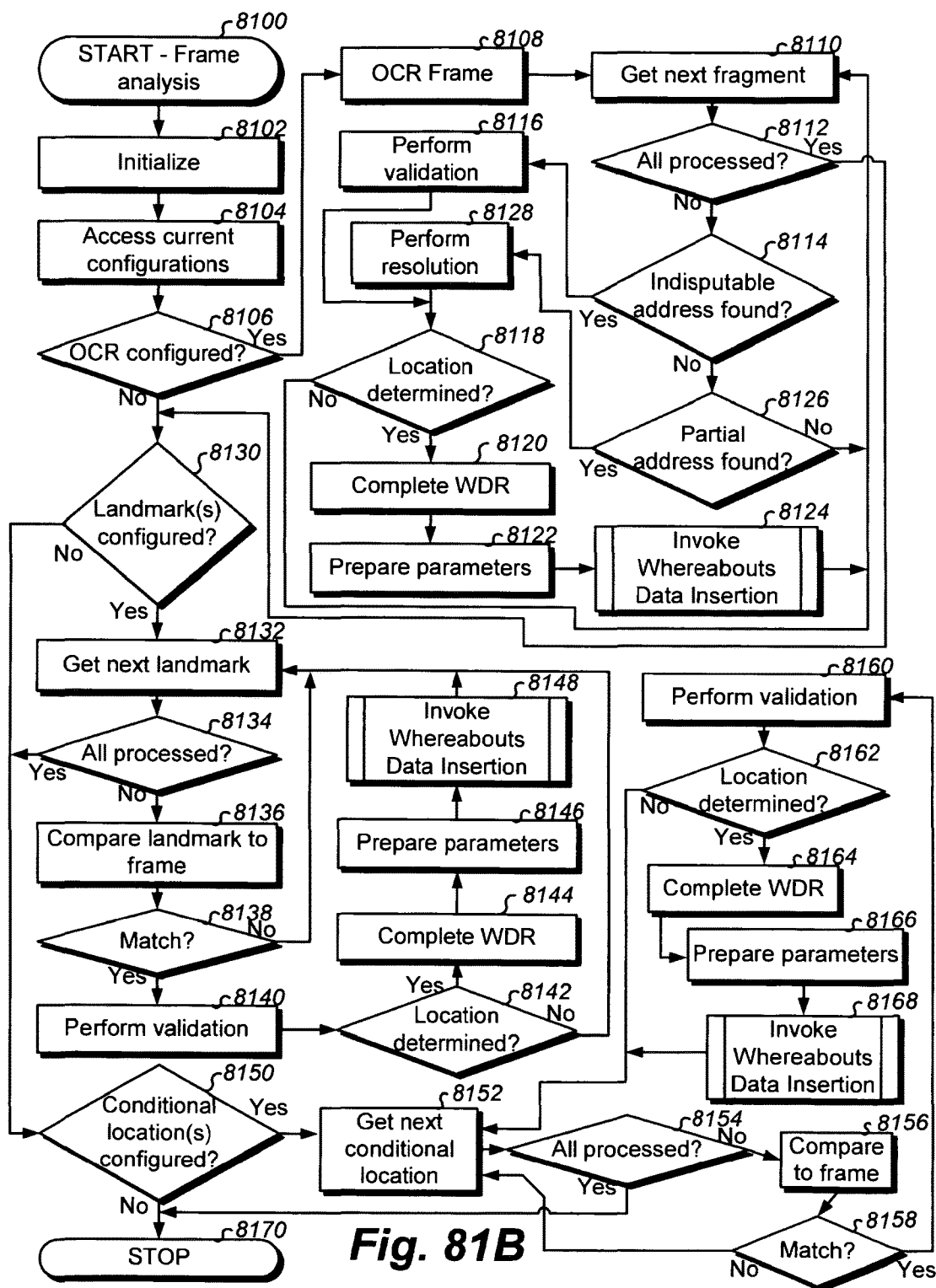
Fig. 80B-4

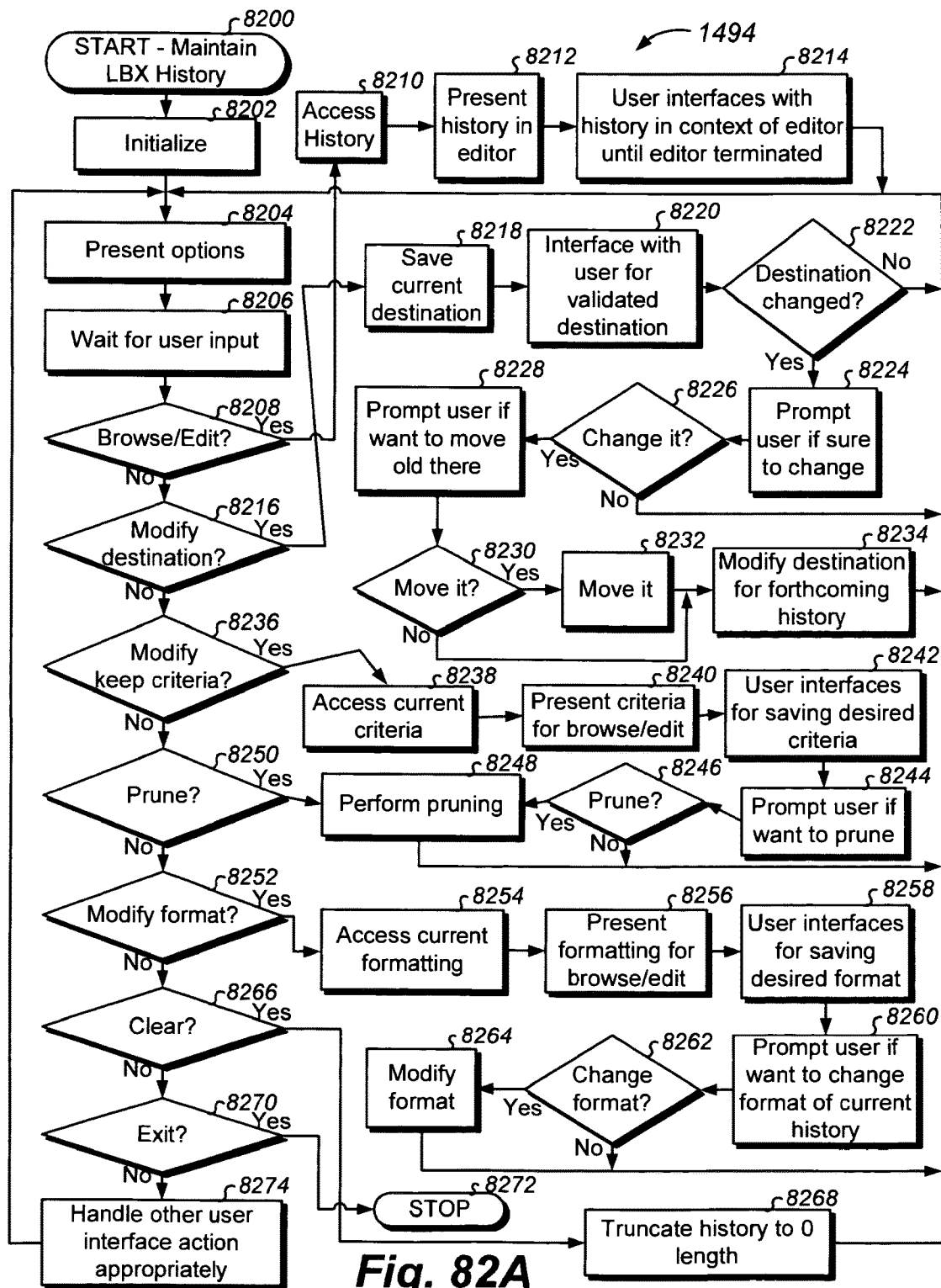
**Fig. 80C**

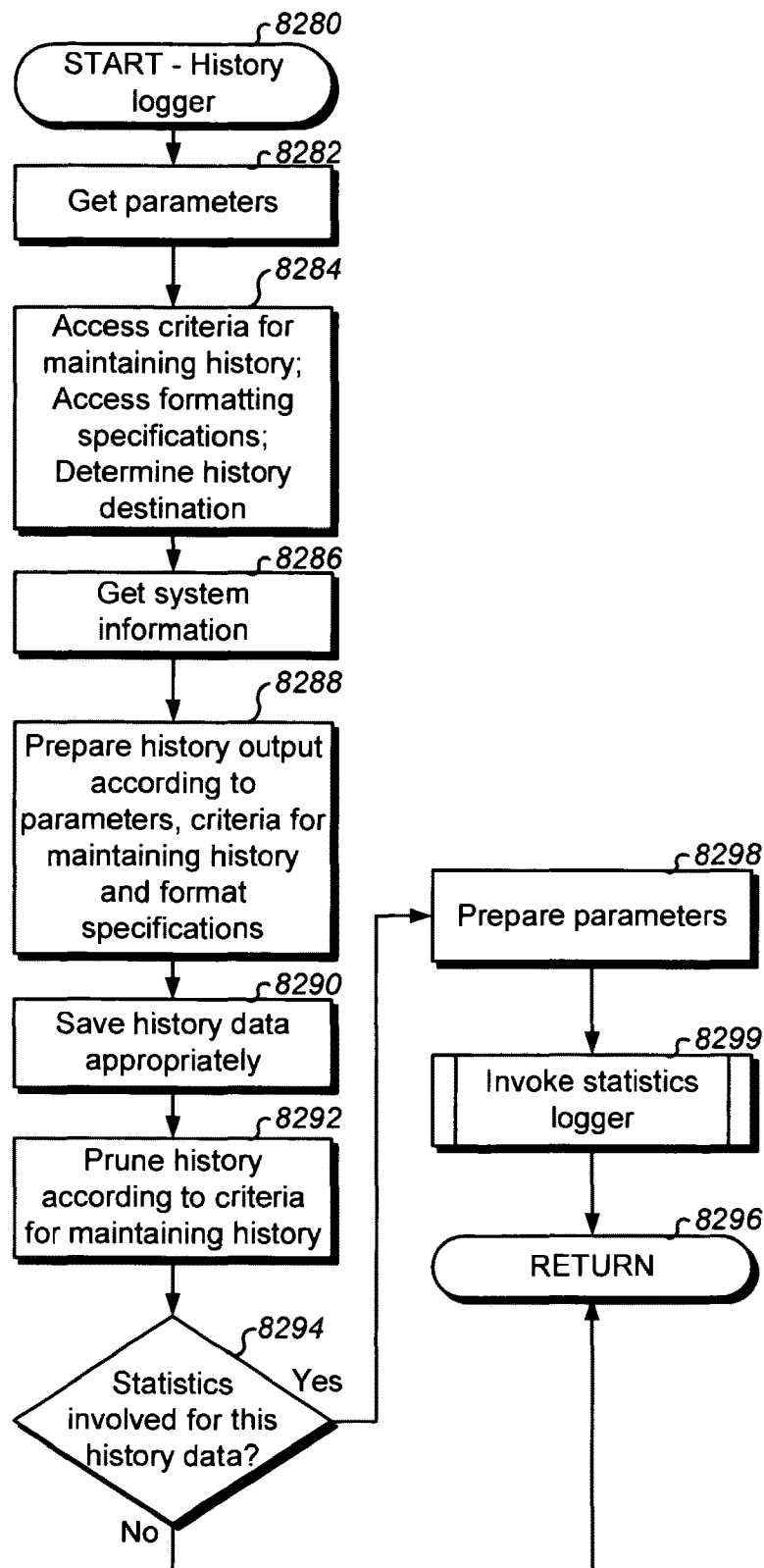
**Fig. 80D**

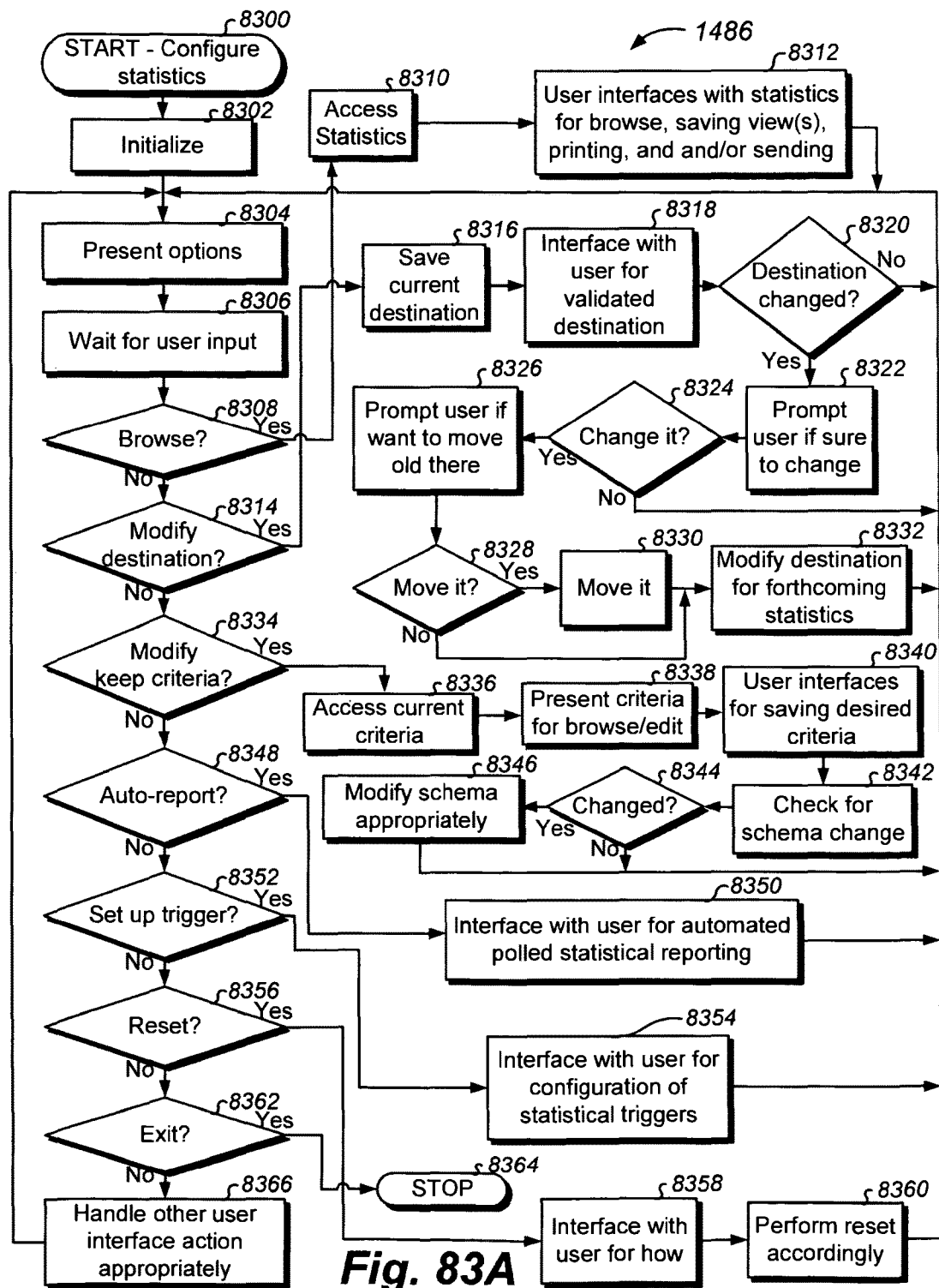
**Fig. 80E**

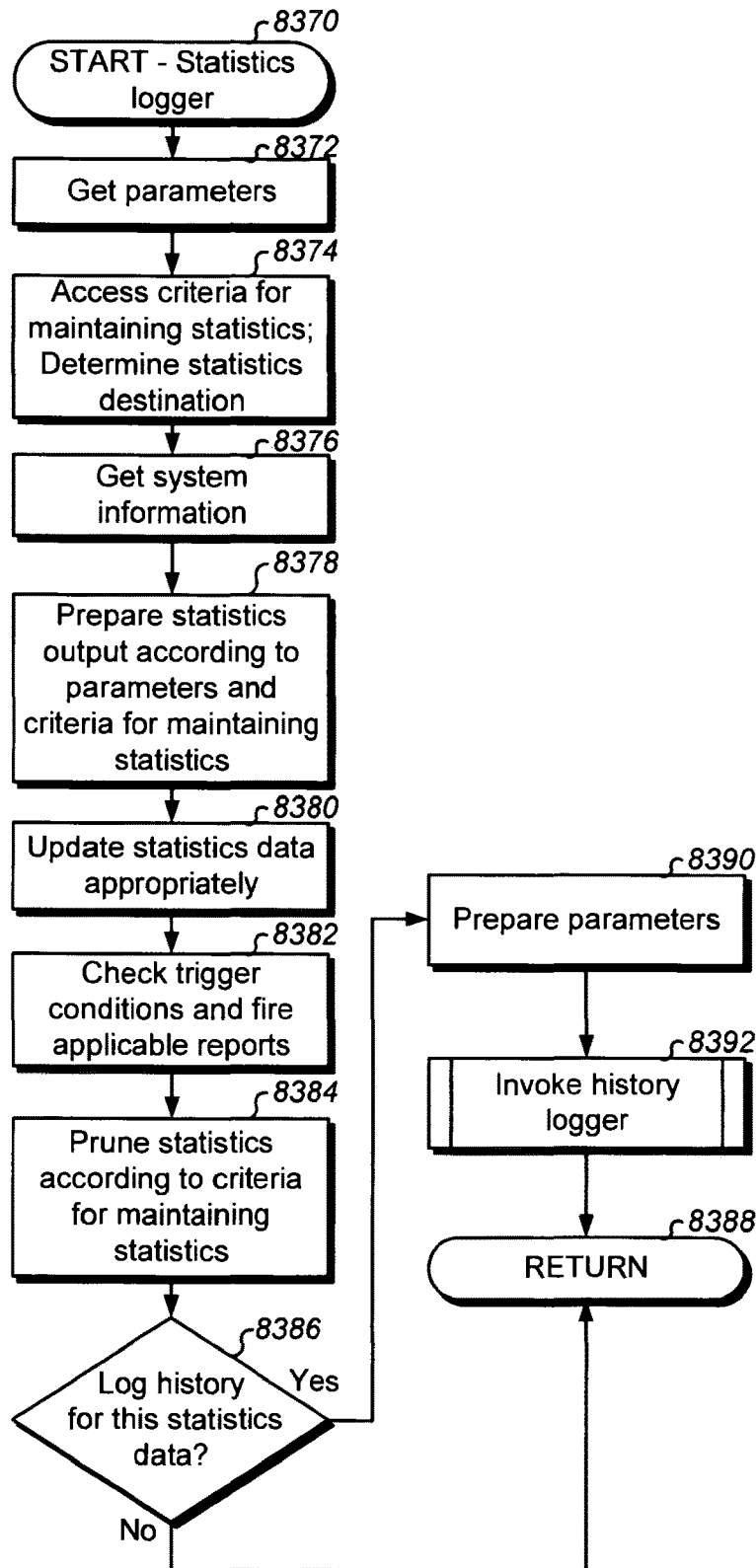
**Fig. 81A**





**Fig. 82B**



**Fig. 83B**

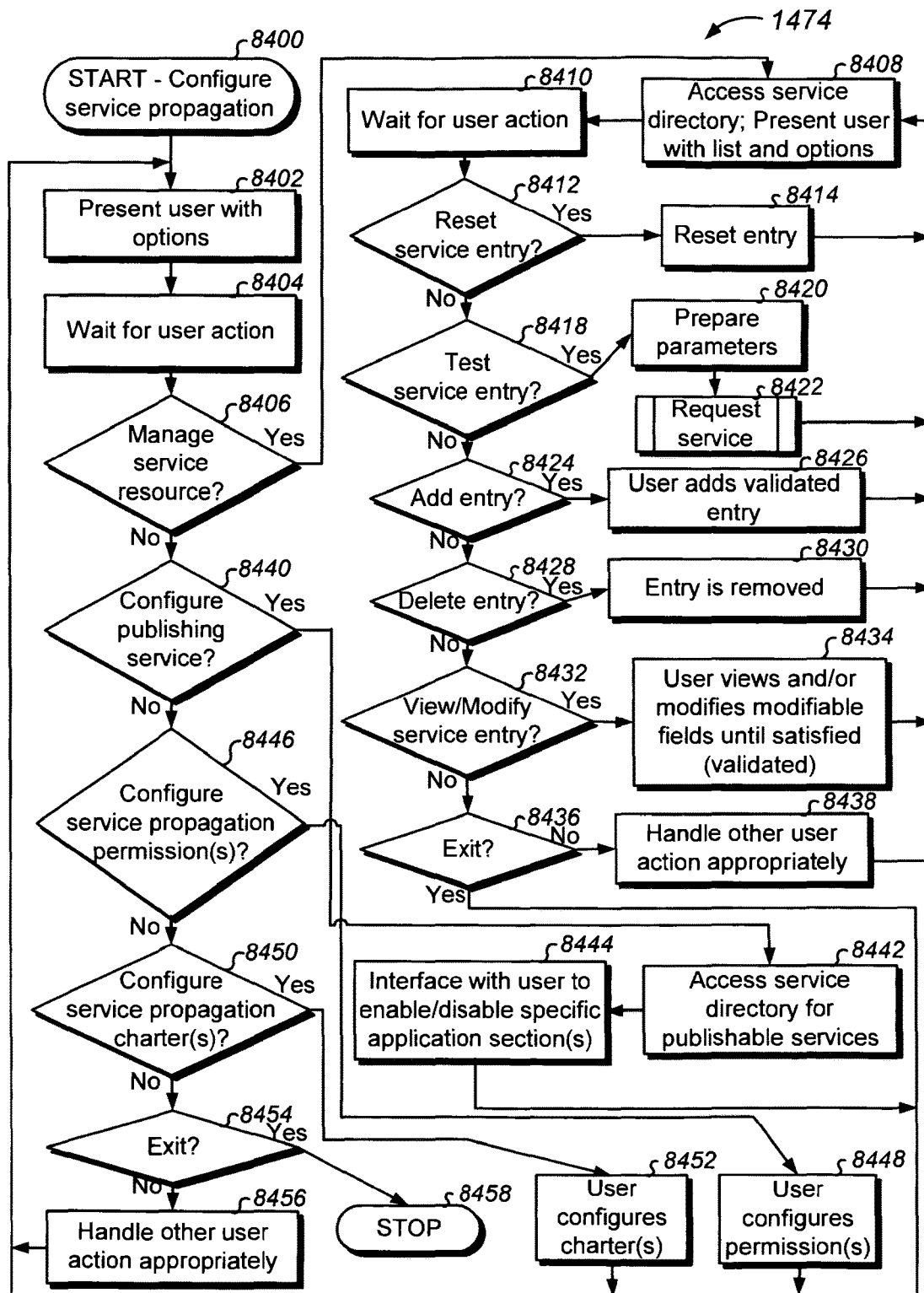
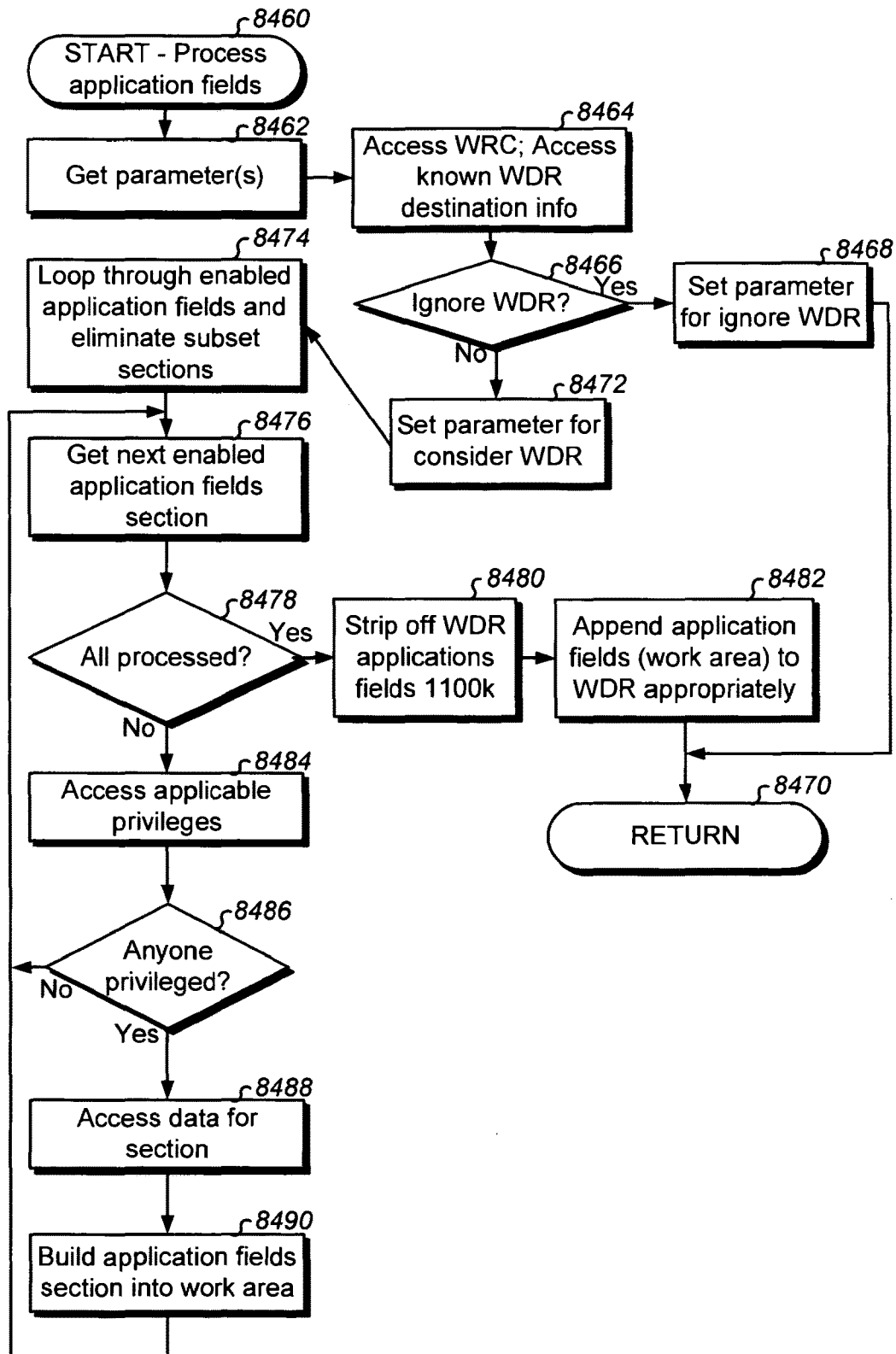
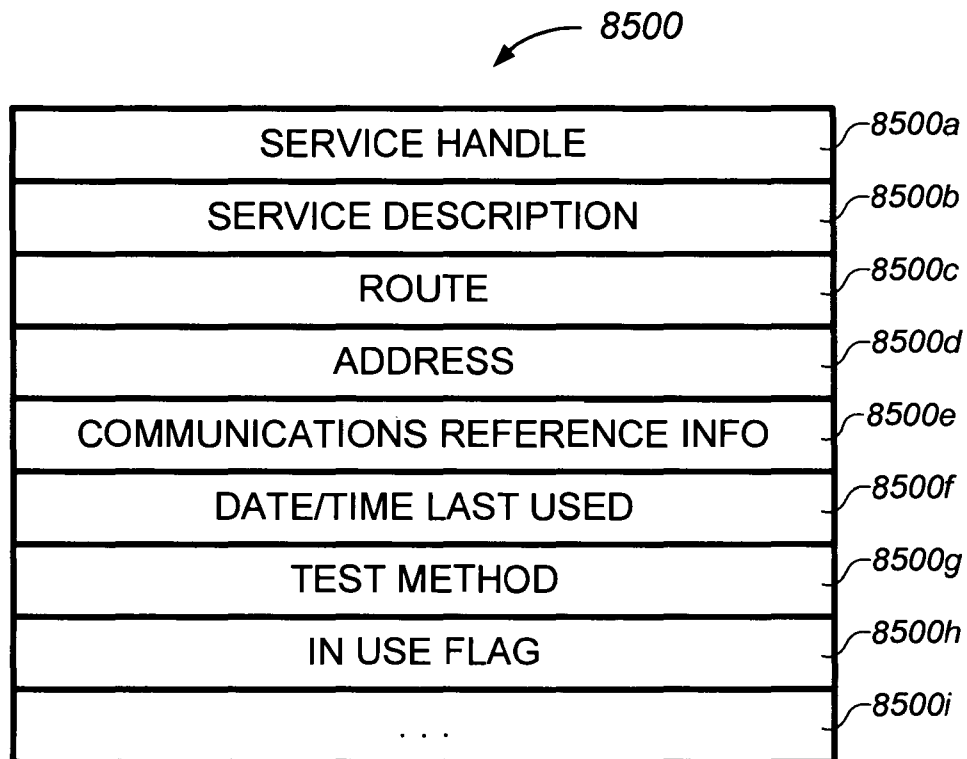
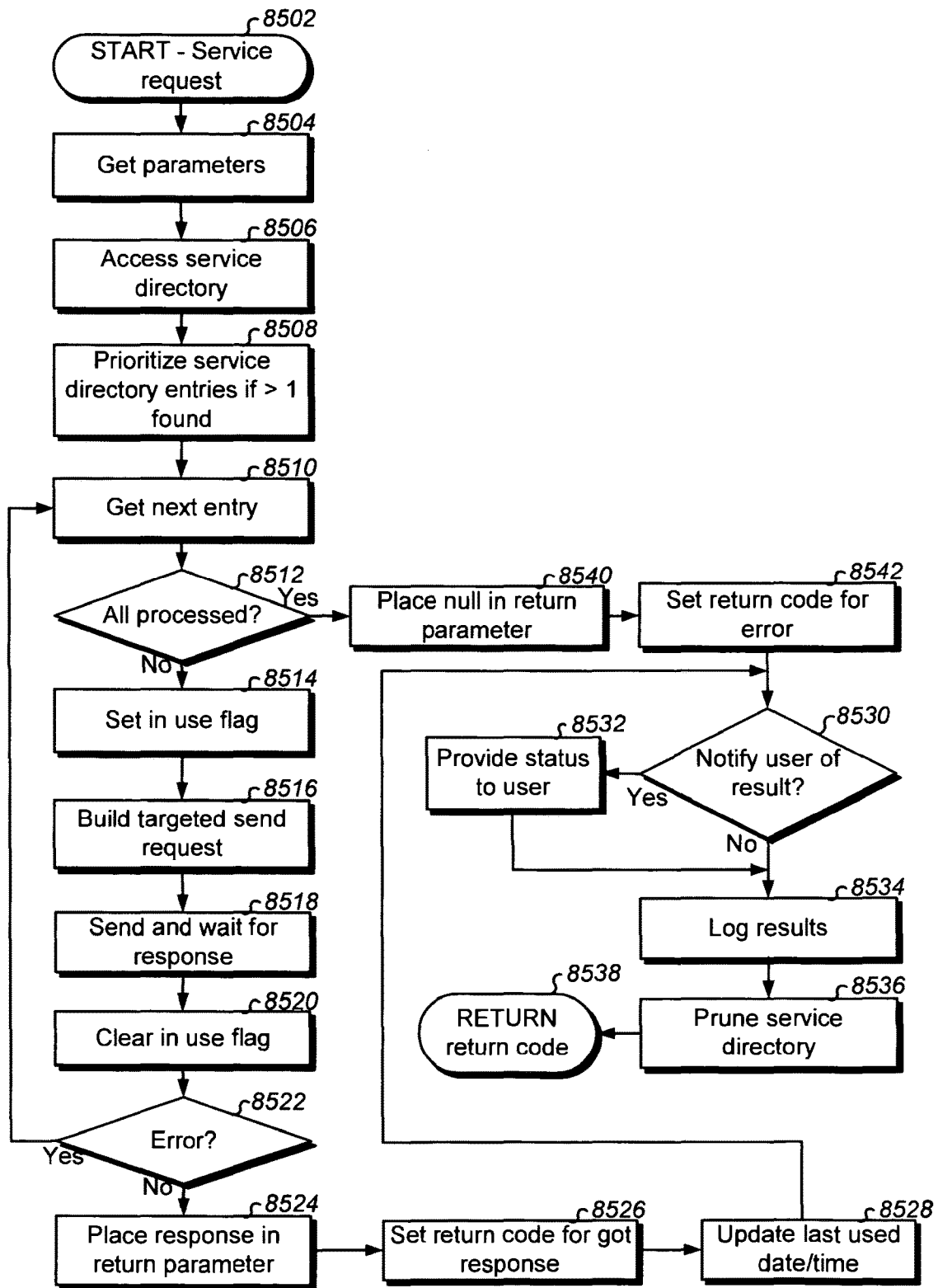
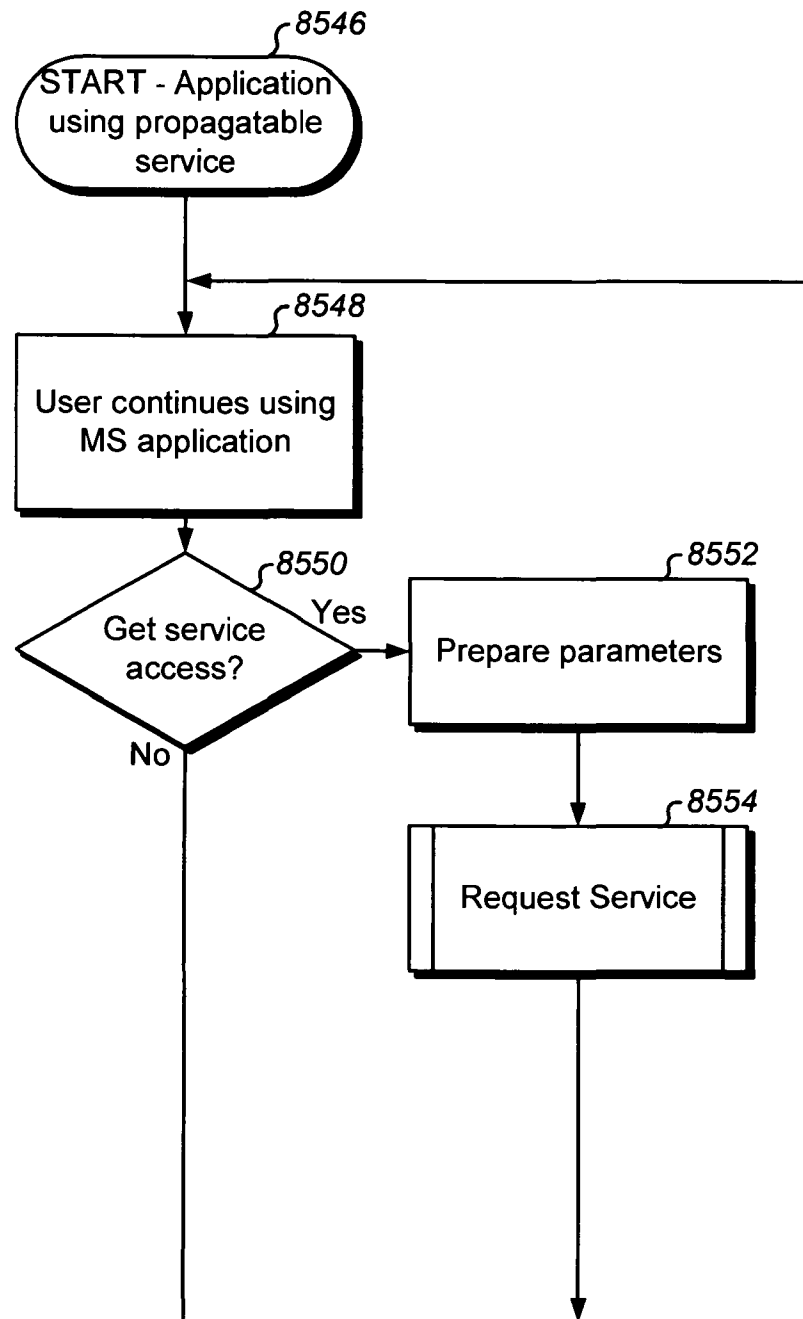


Fig. 84A

**Fig. 84B**

***Fig. 85A***

**Fig. 85B**

**Fig. 85C**

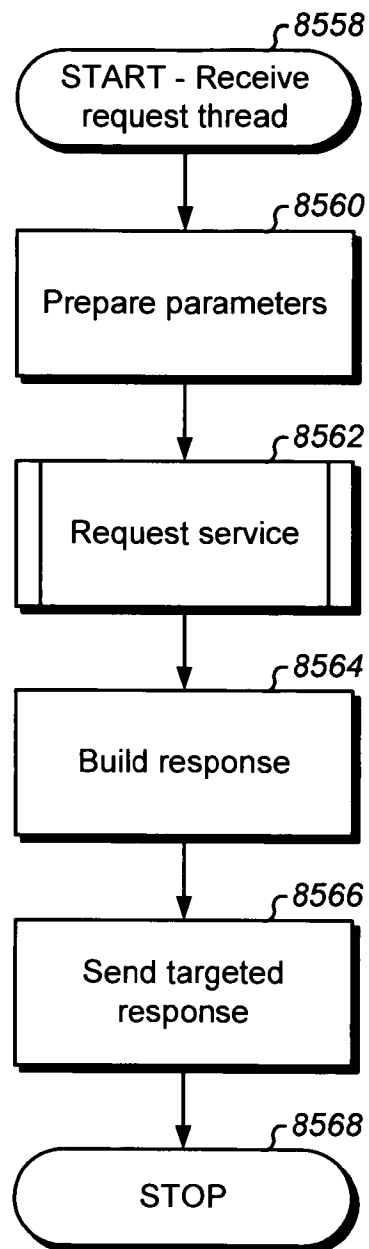
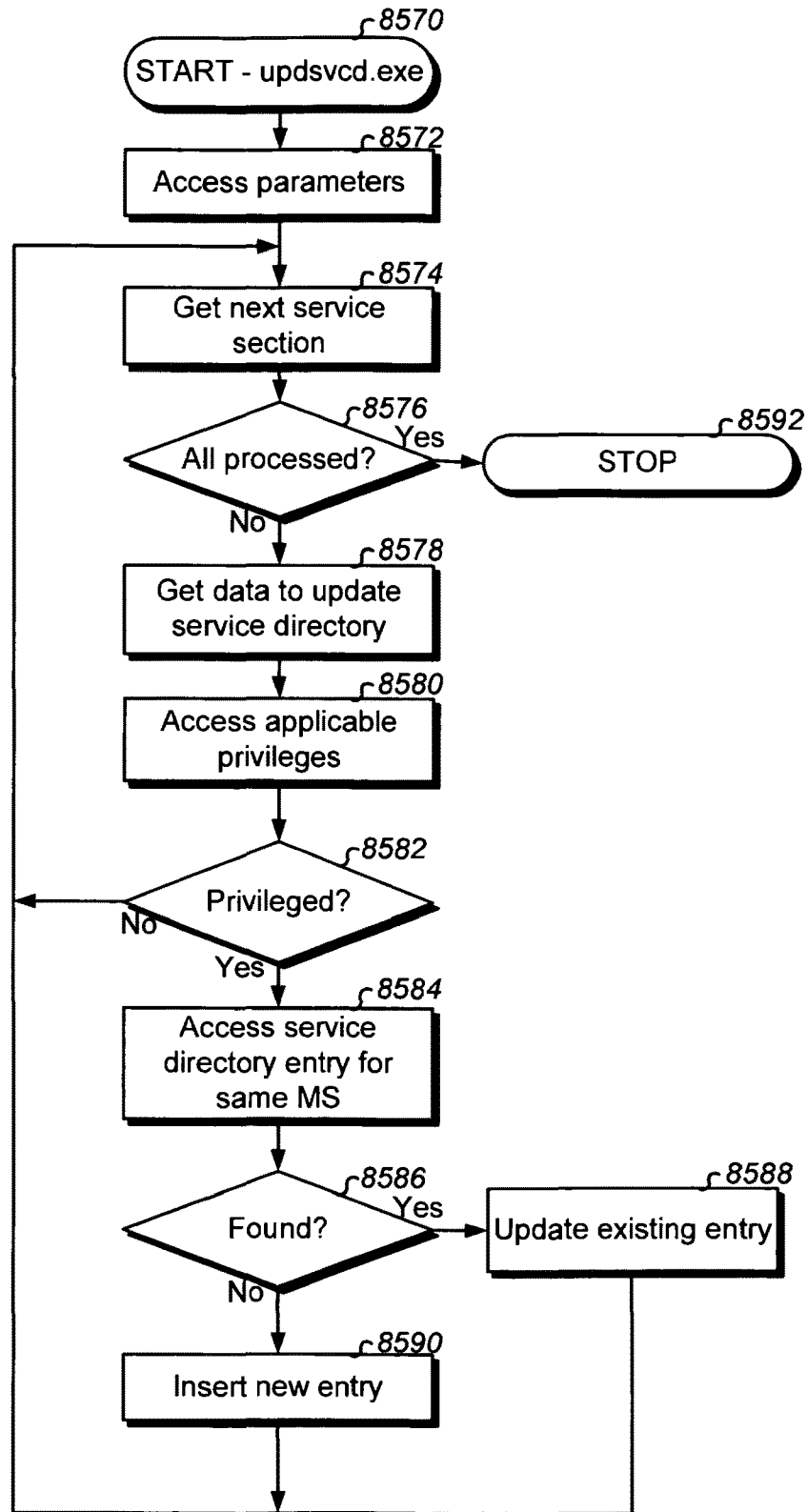
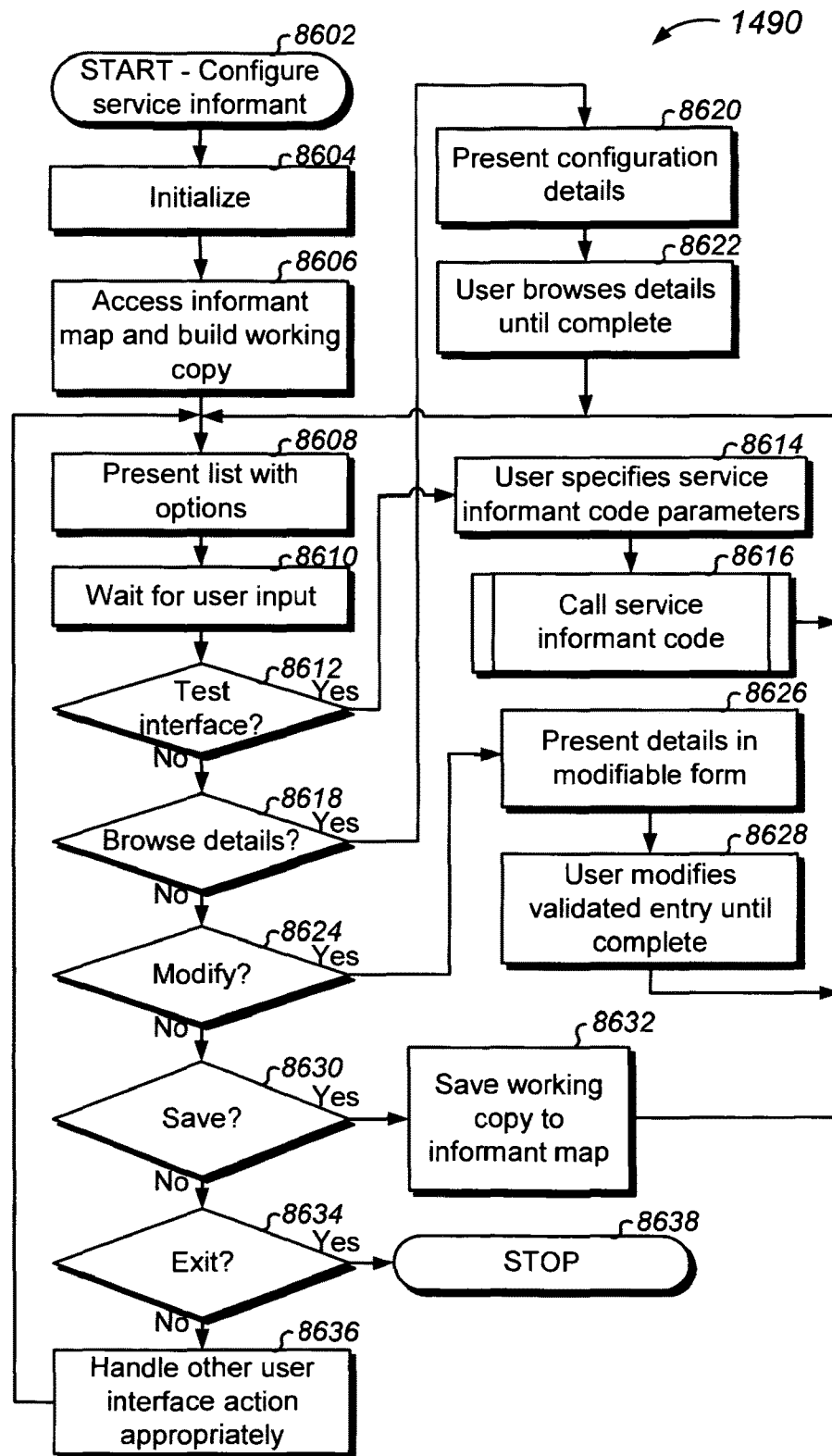
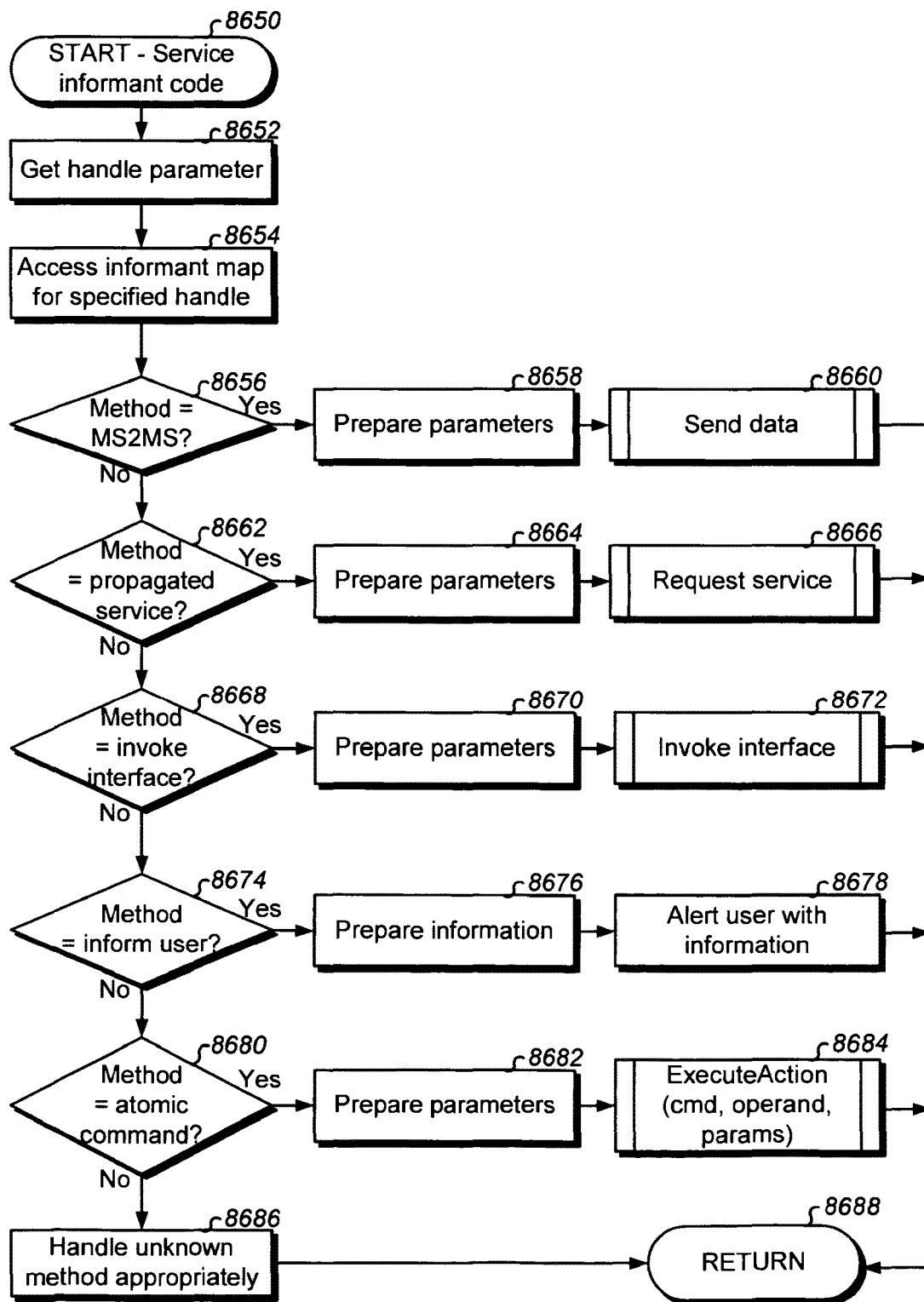


Fig. 85D

**Fig. 85E**

**Fig. 86A**

**Fig. 86B**

U.S. Patent

Nov. 19, 2019

Sheet 297 of 322

US 10,477,994 B2

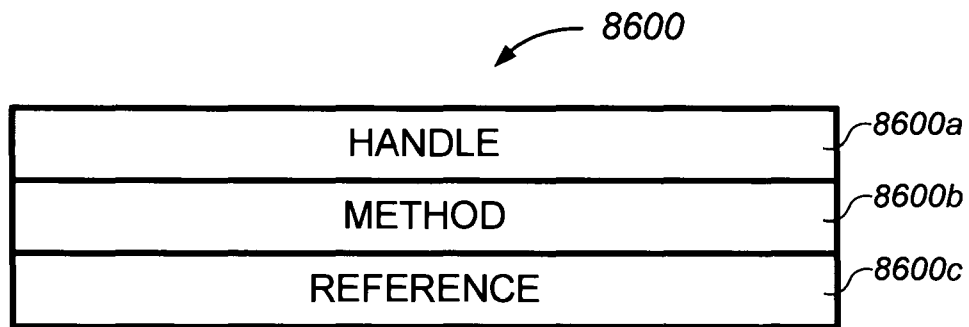


Fig. 86C

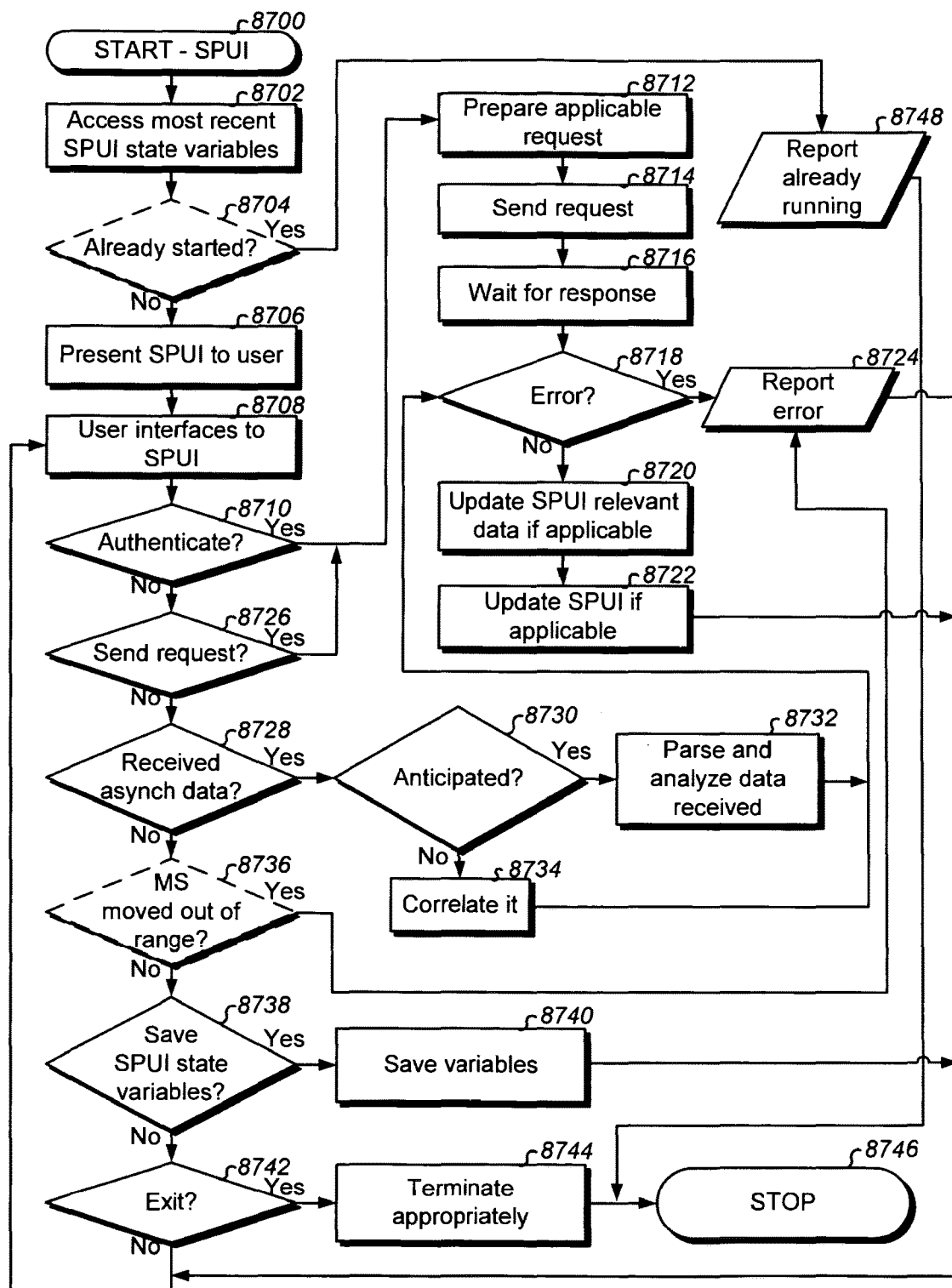


Fig. 87A

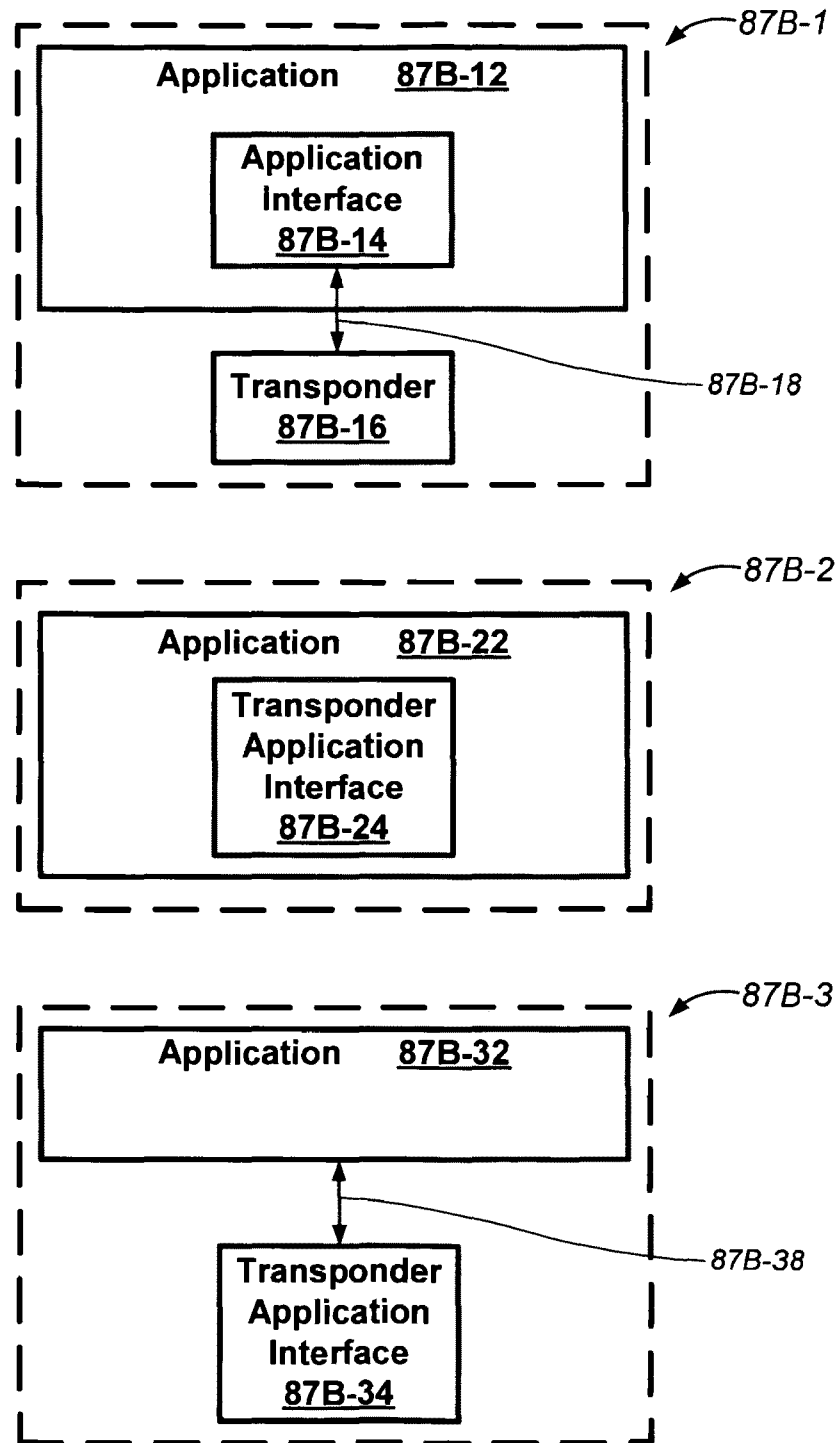
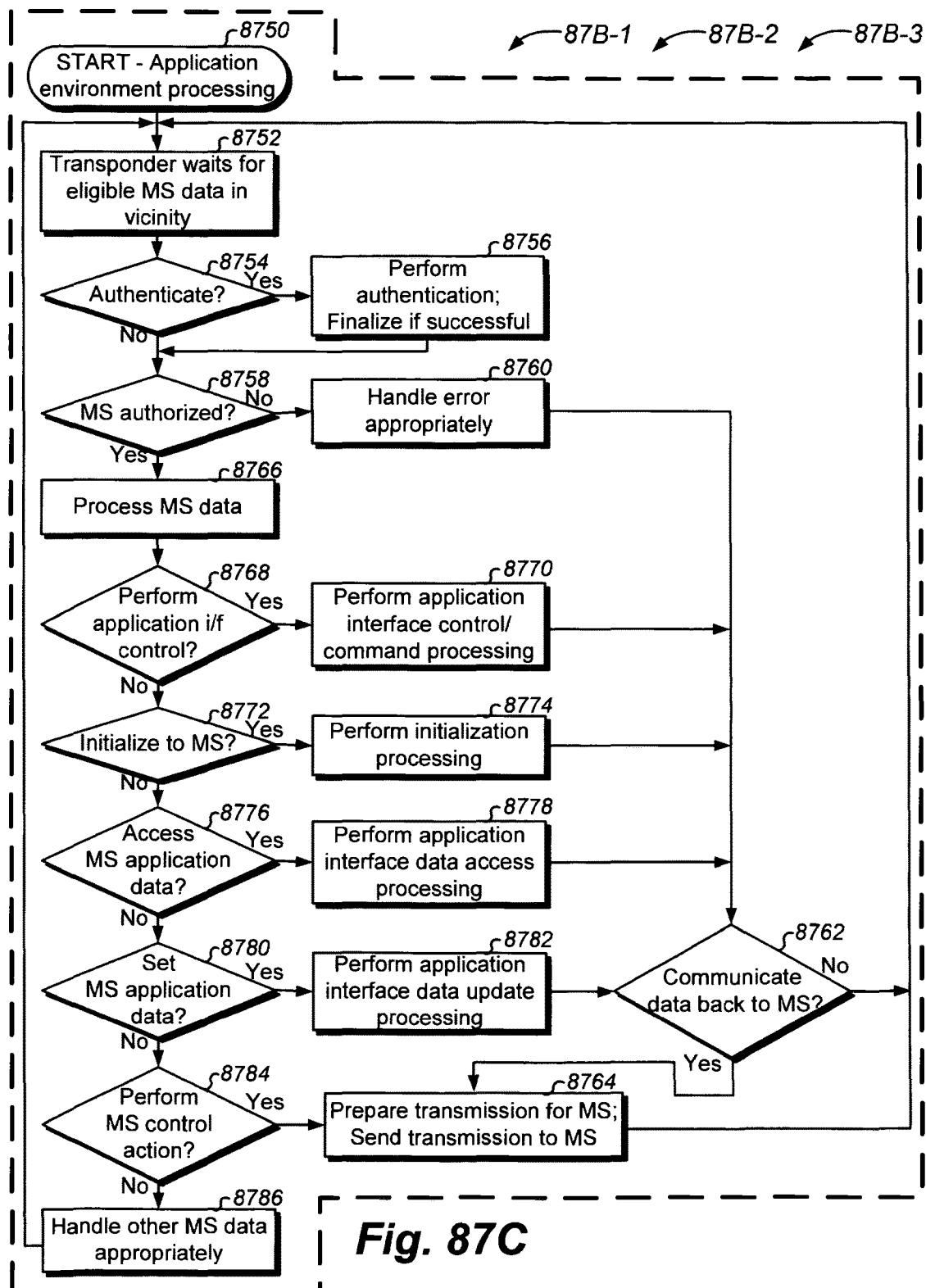
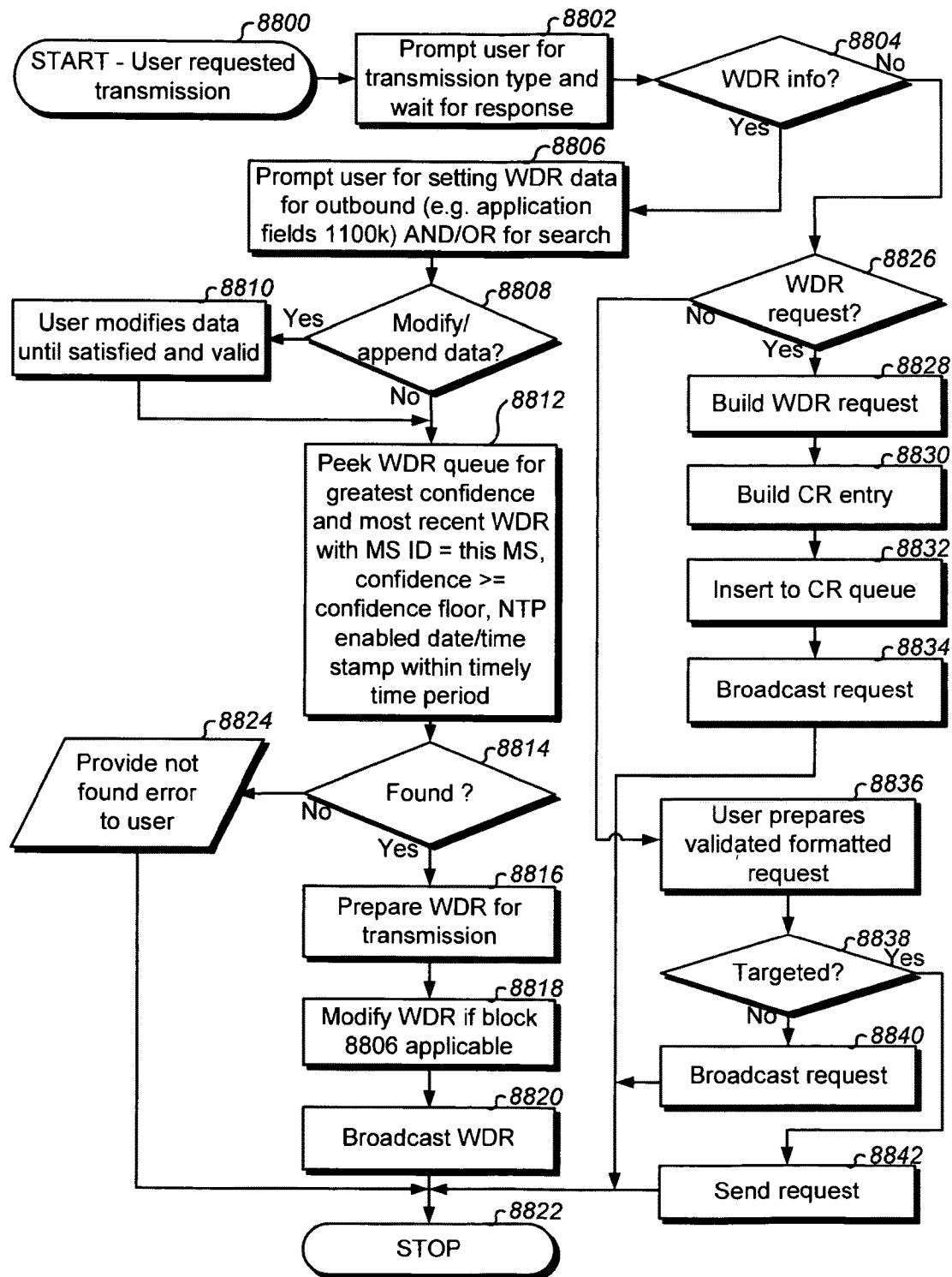
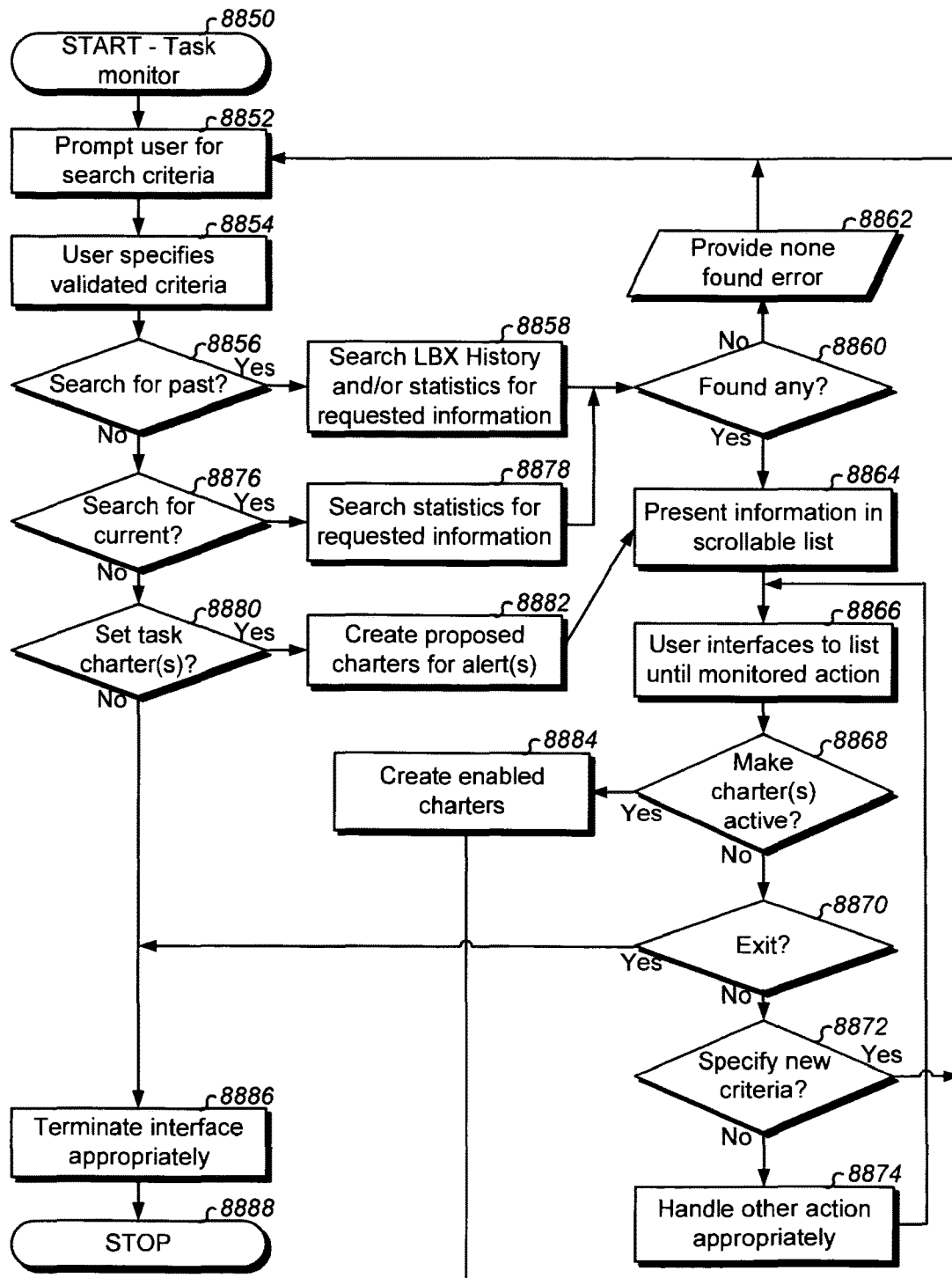
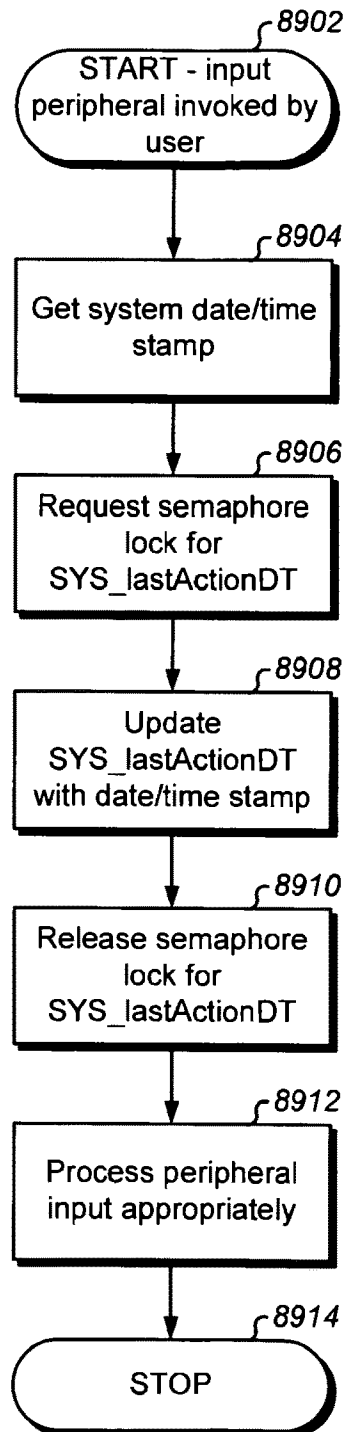


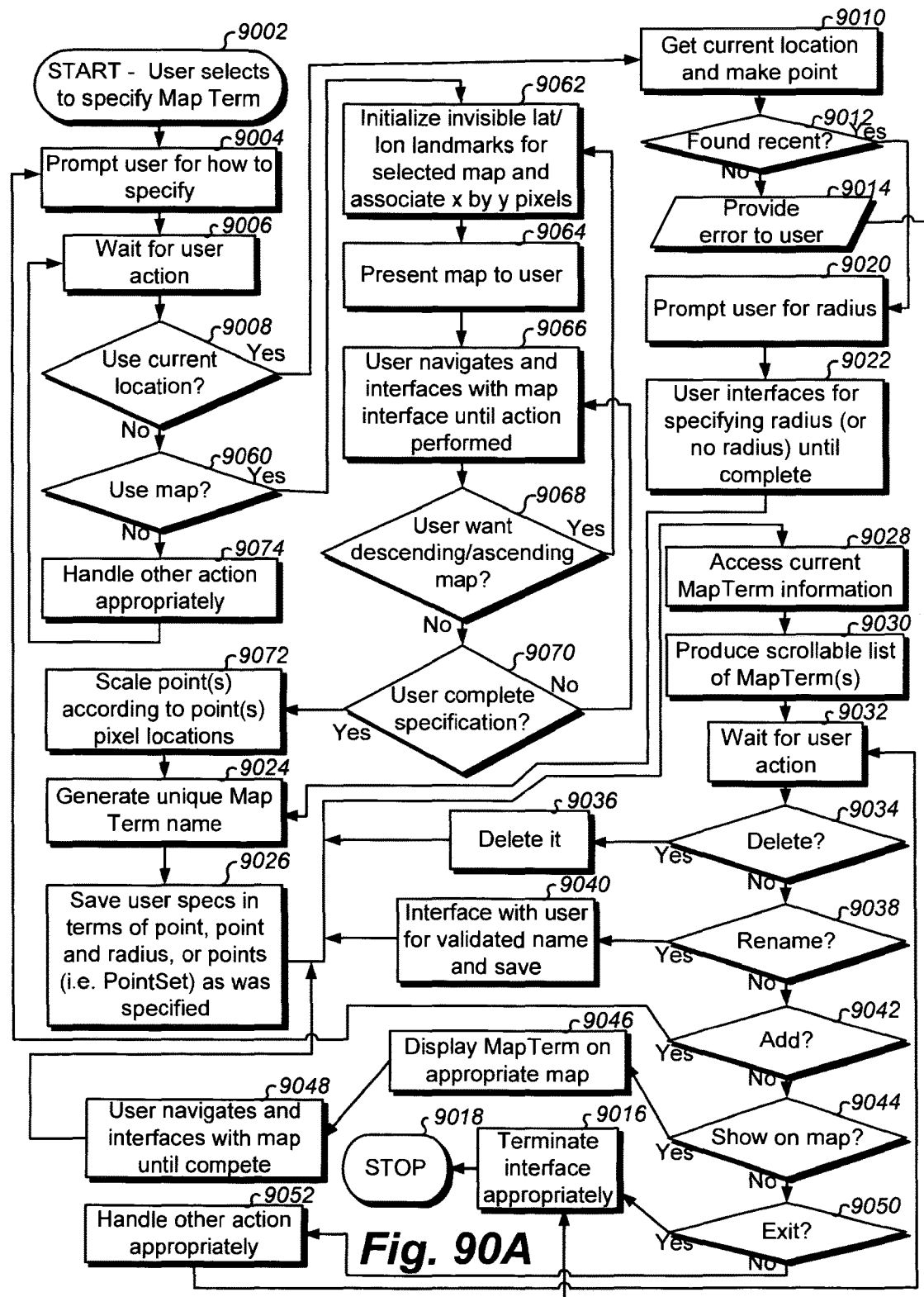
Fig. 87B



**Fig. 88A**

**Fig. 88B**

**Fig. 89A**



U.S. Patent

Nov. 19, 2019

Sheet 305 of 322

US 10,477,994 B2

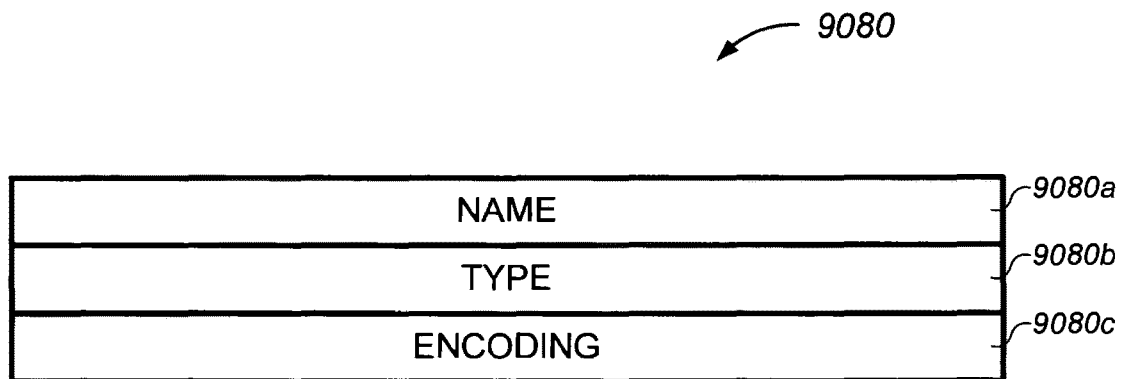
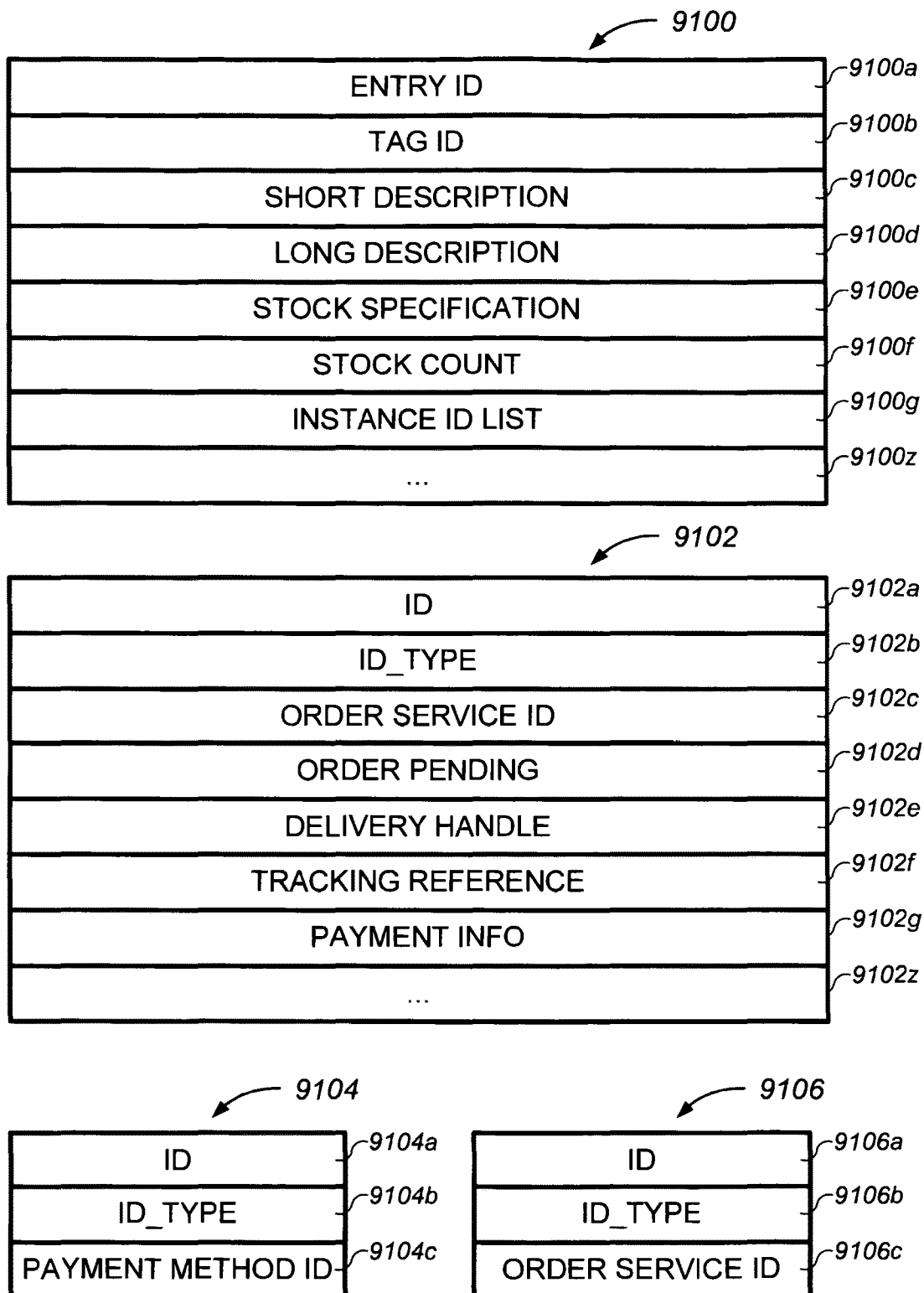
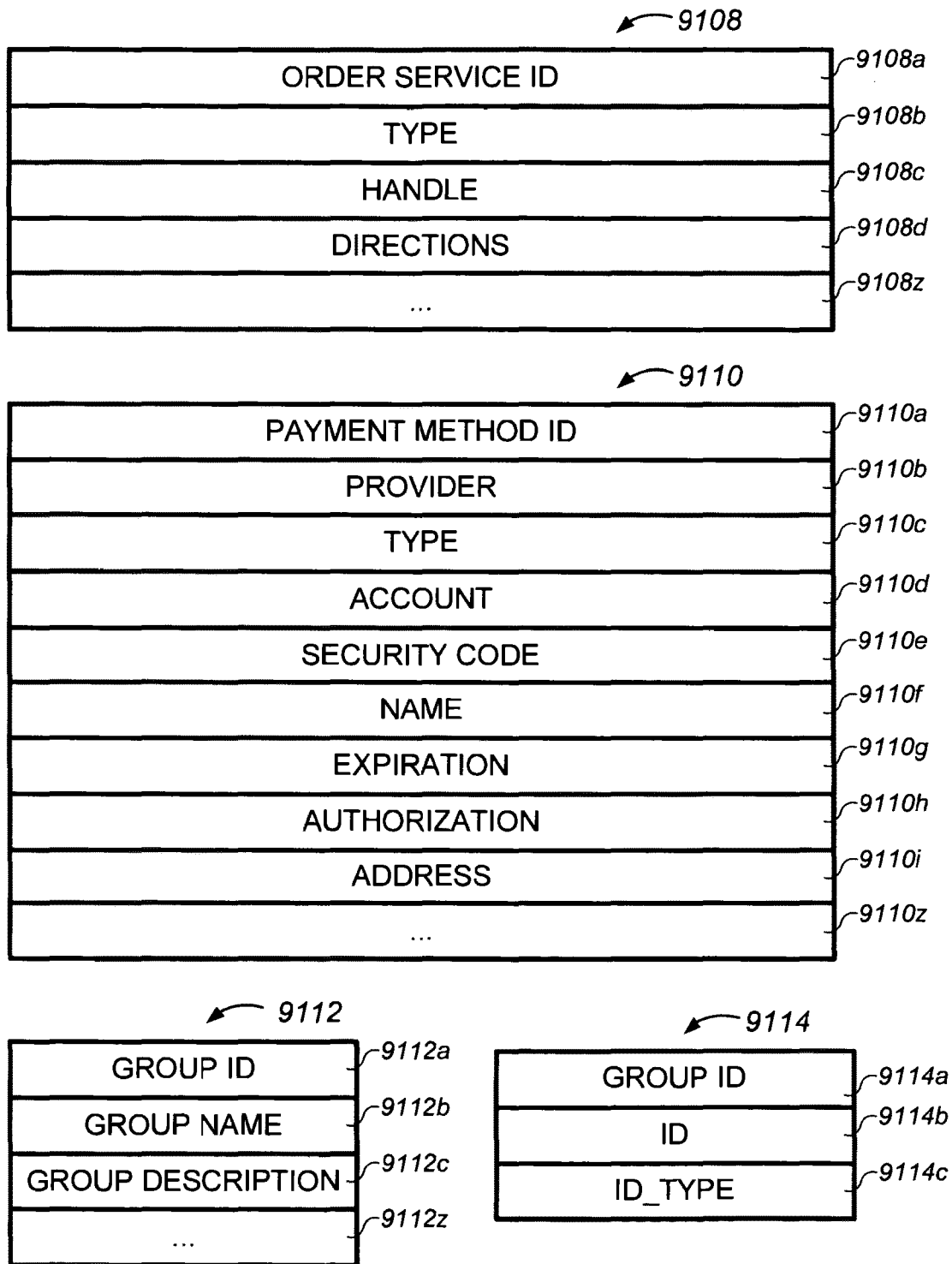
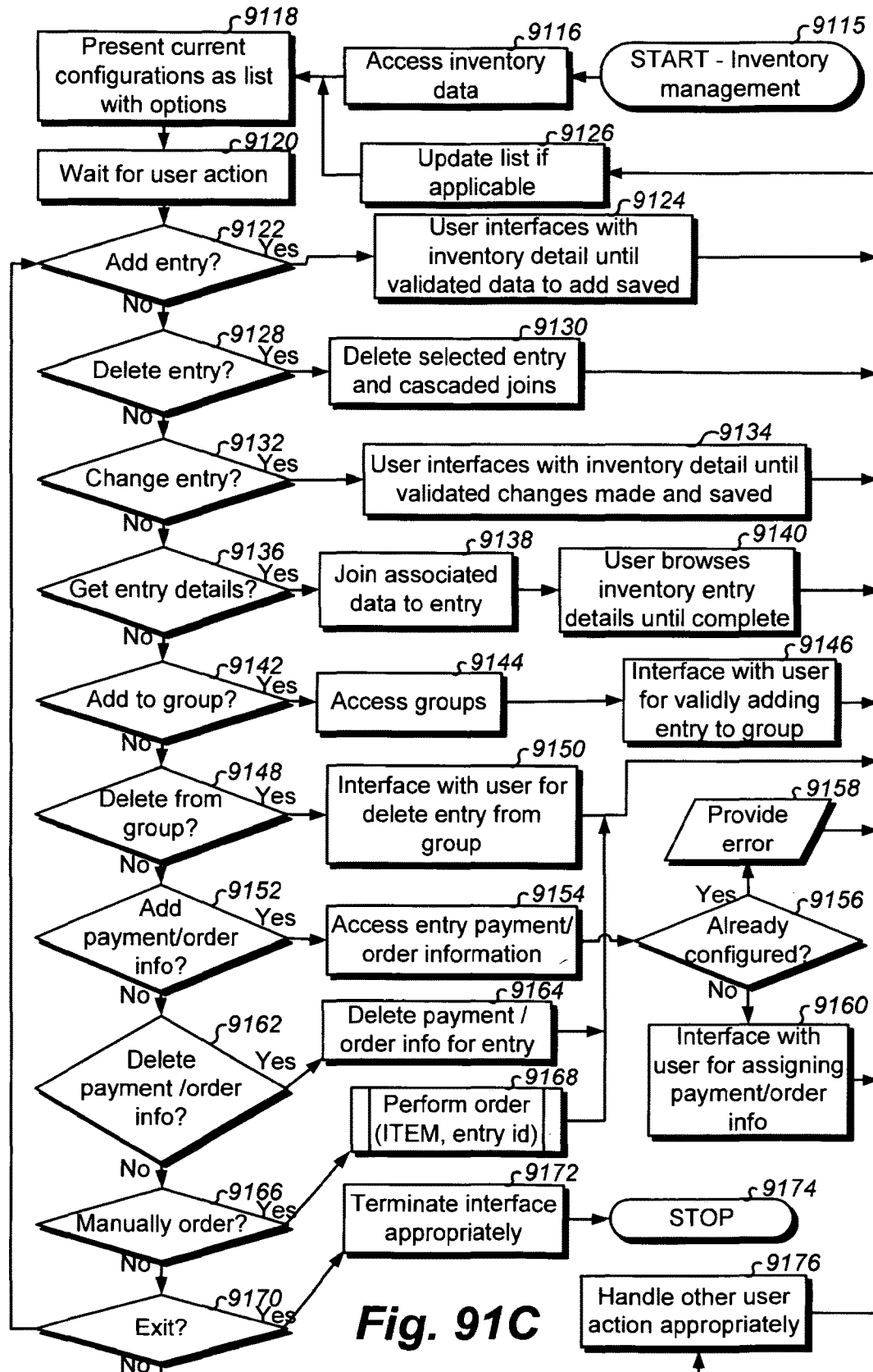
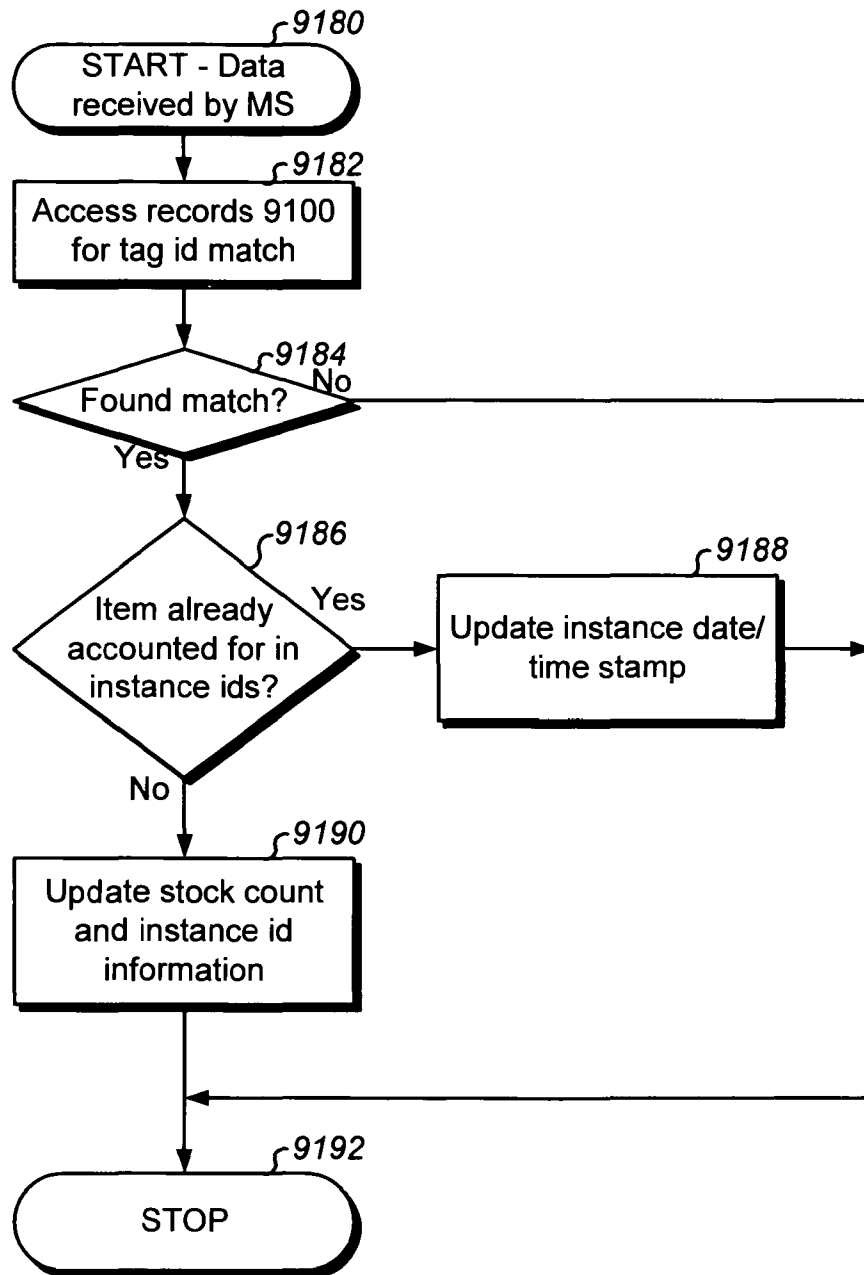


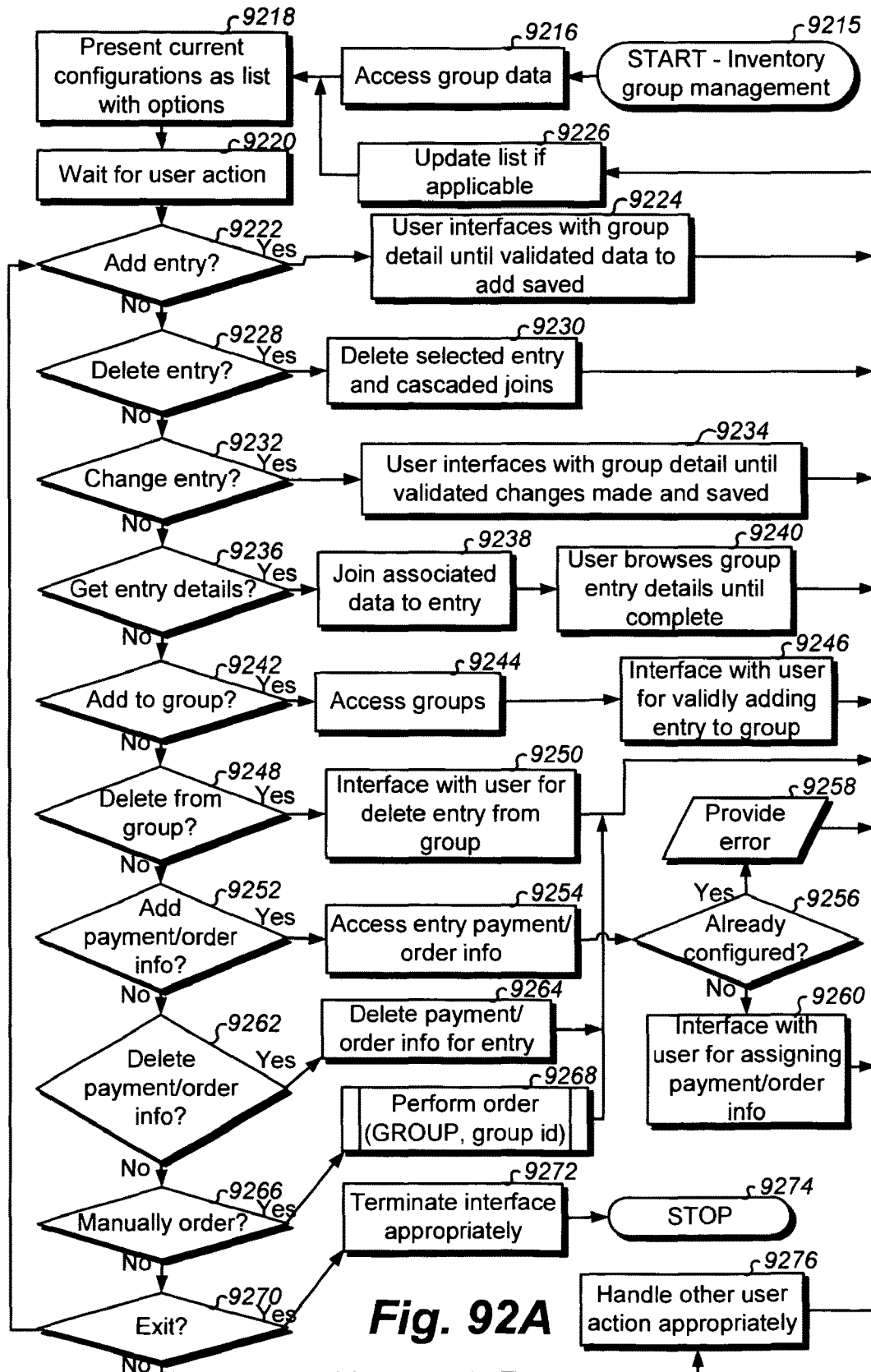
Fig. 90B

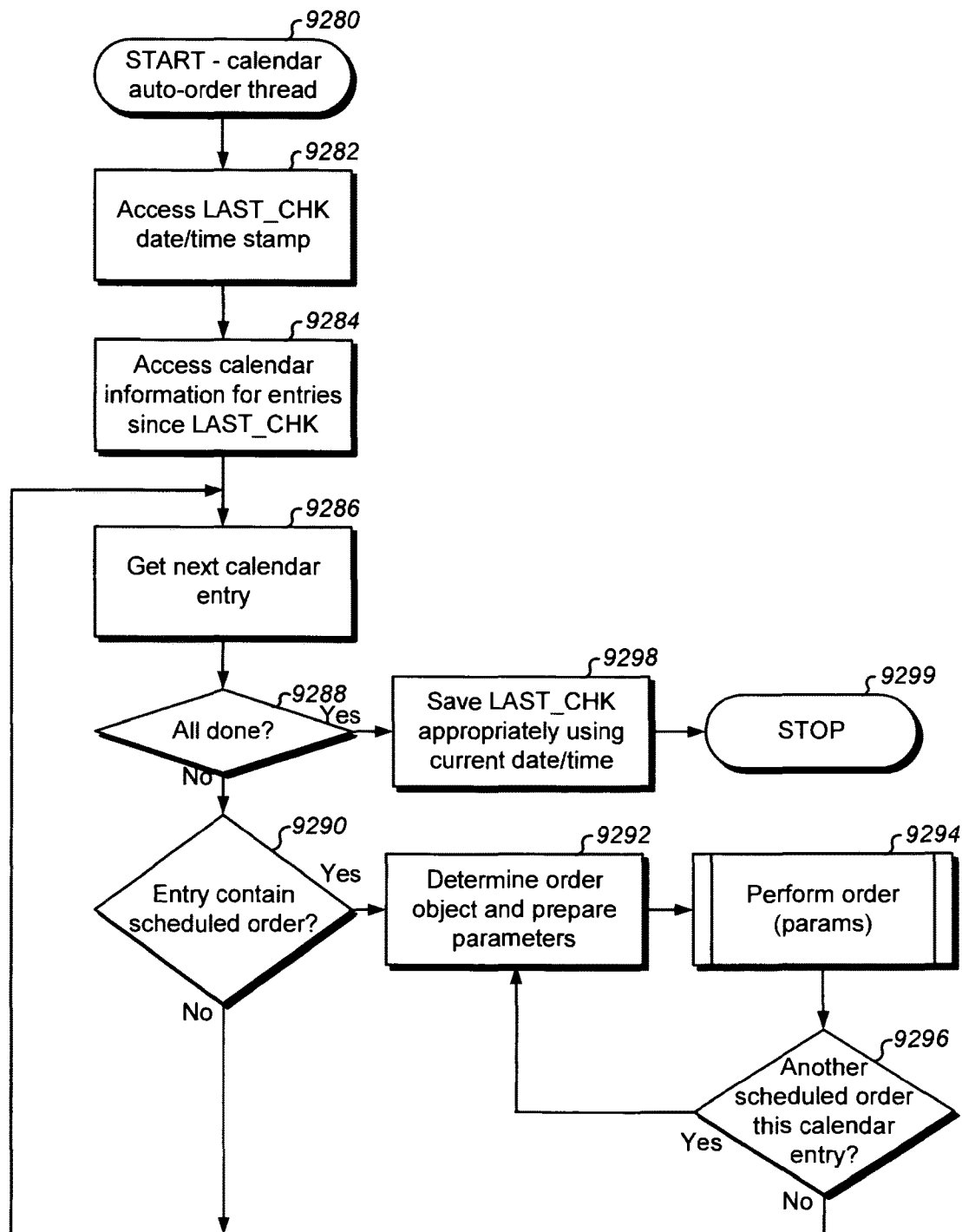
**Fig. 91A**

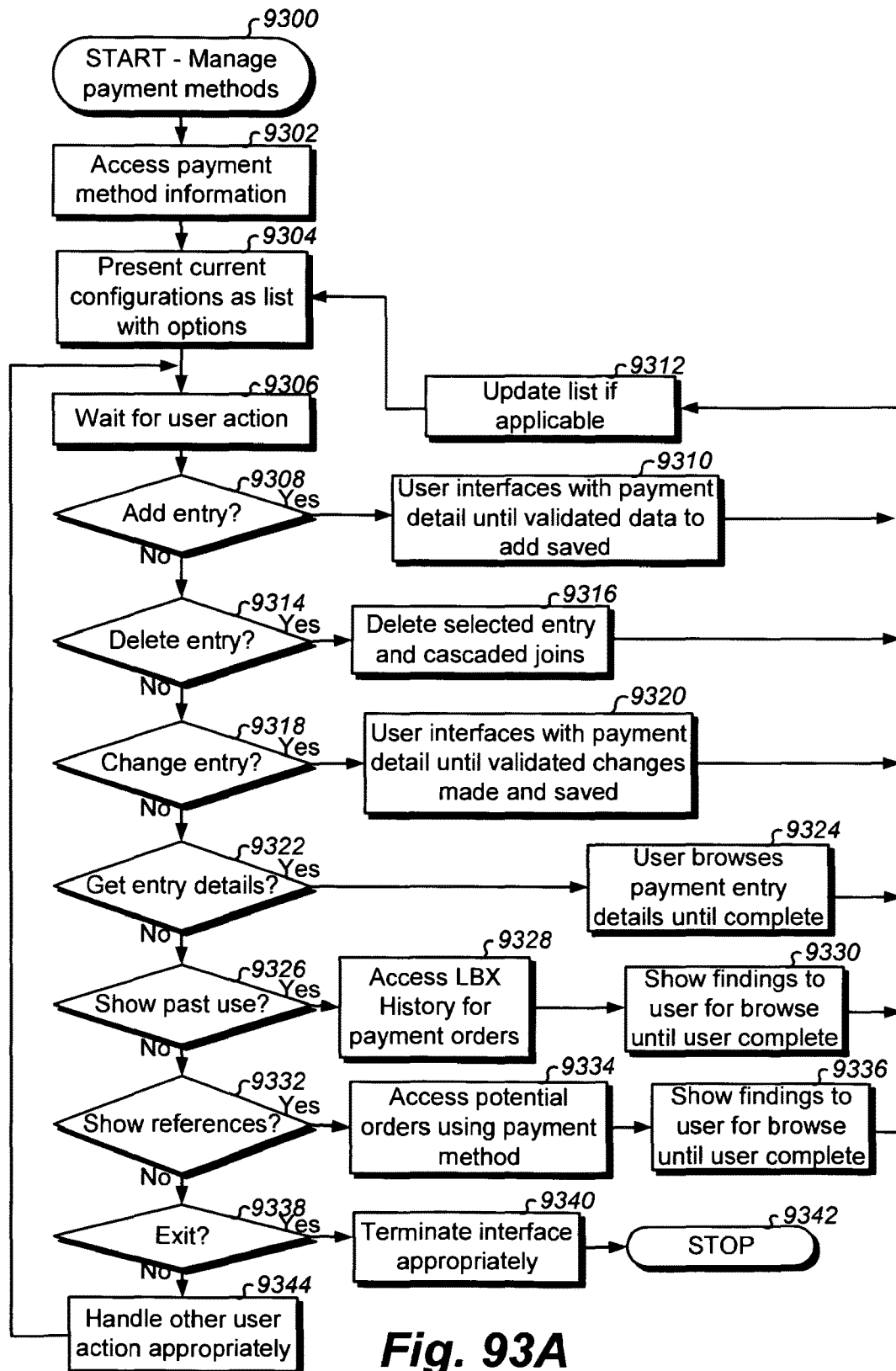
**Fig. 91B**

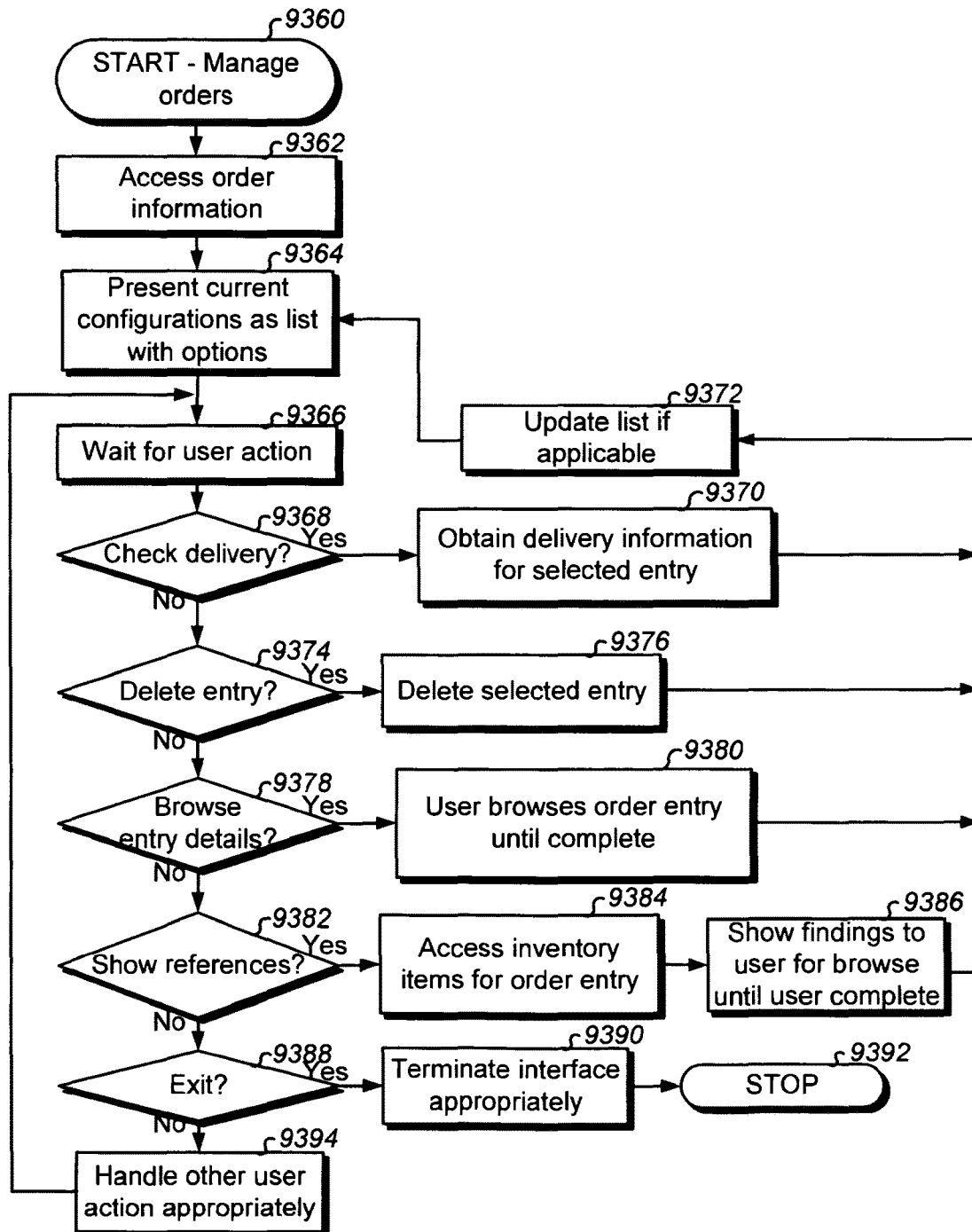


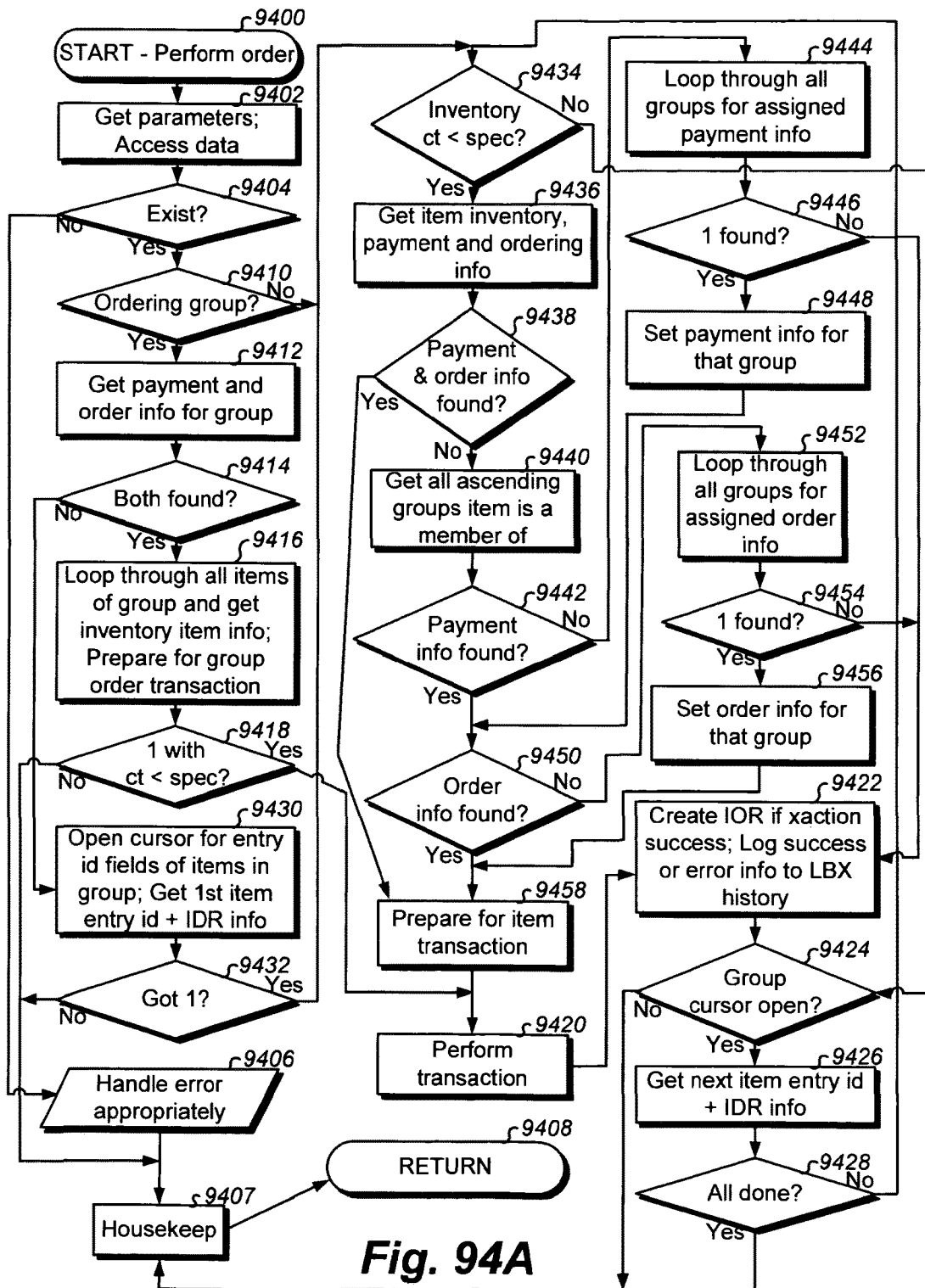
**Fig. 91D**

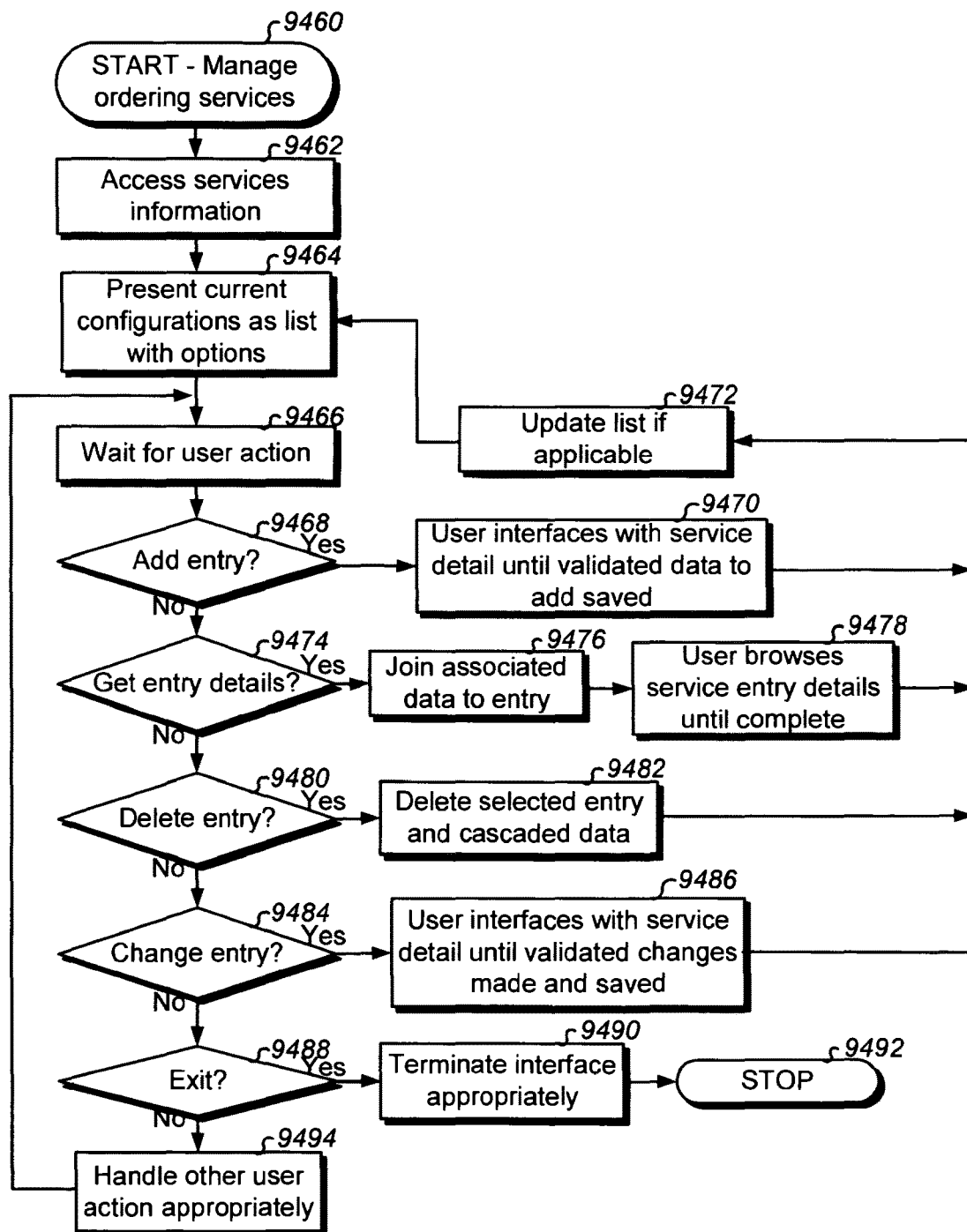


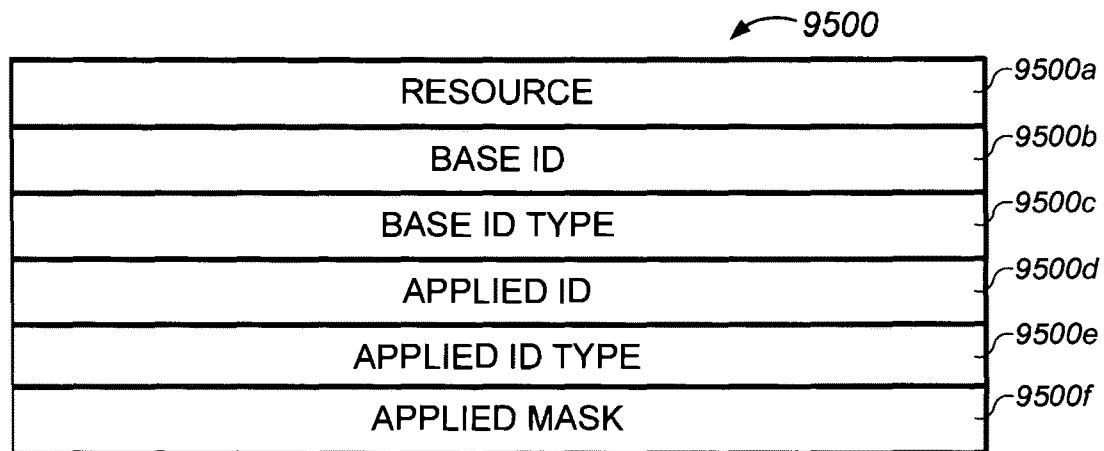
**Fig. 92B**

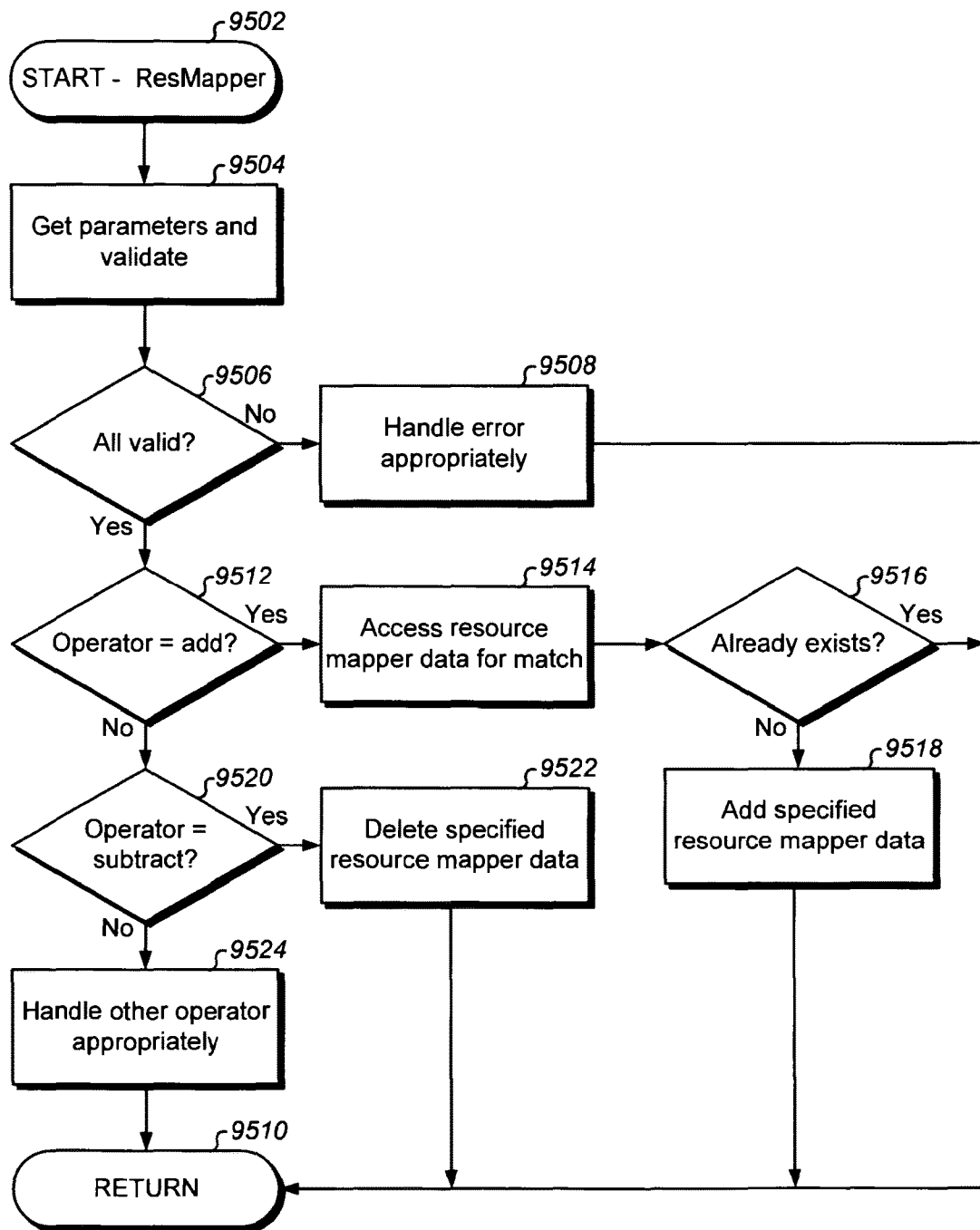


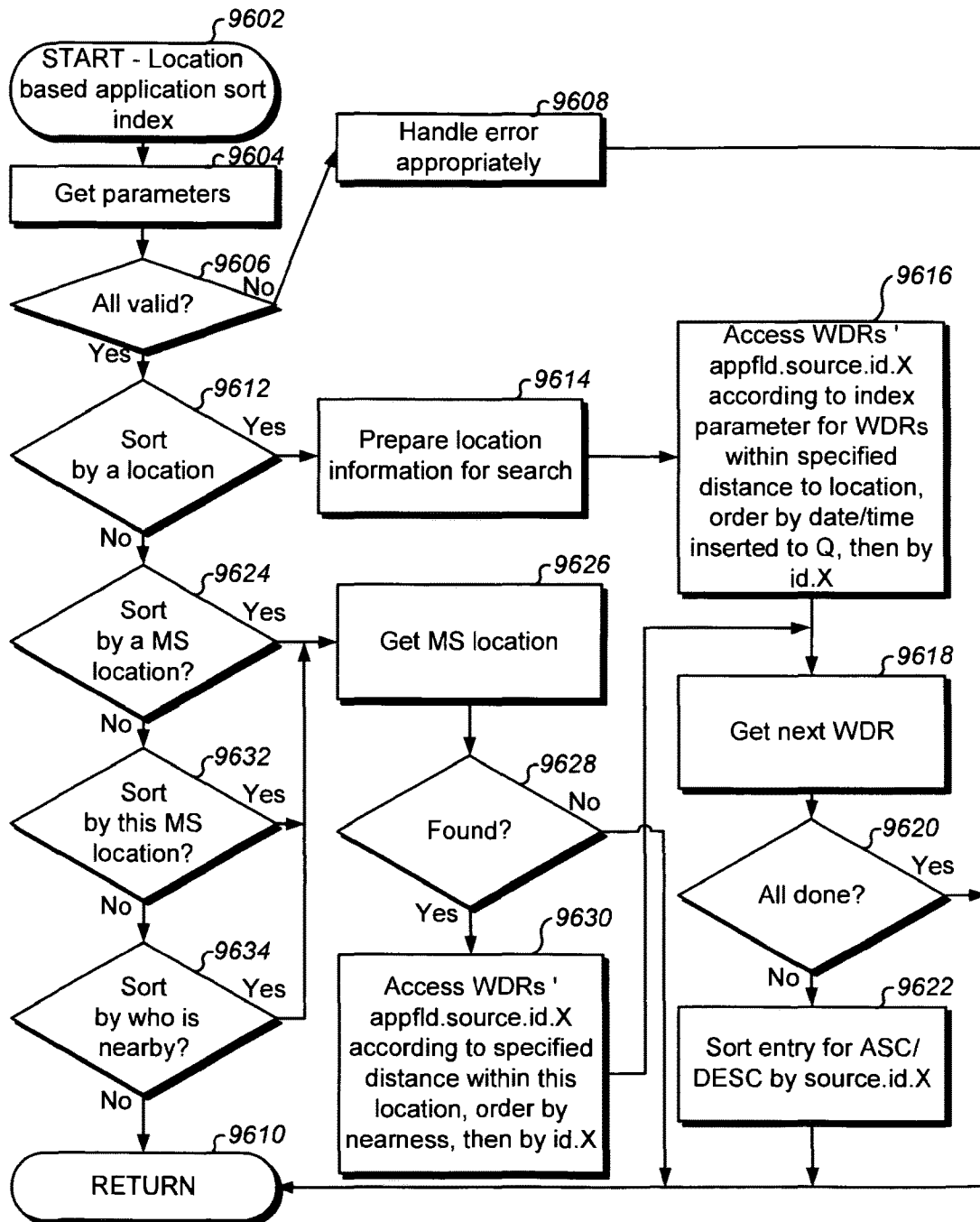
**Fig. 93B**

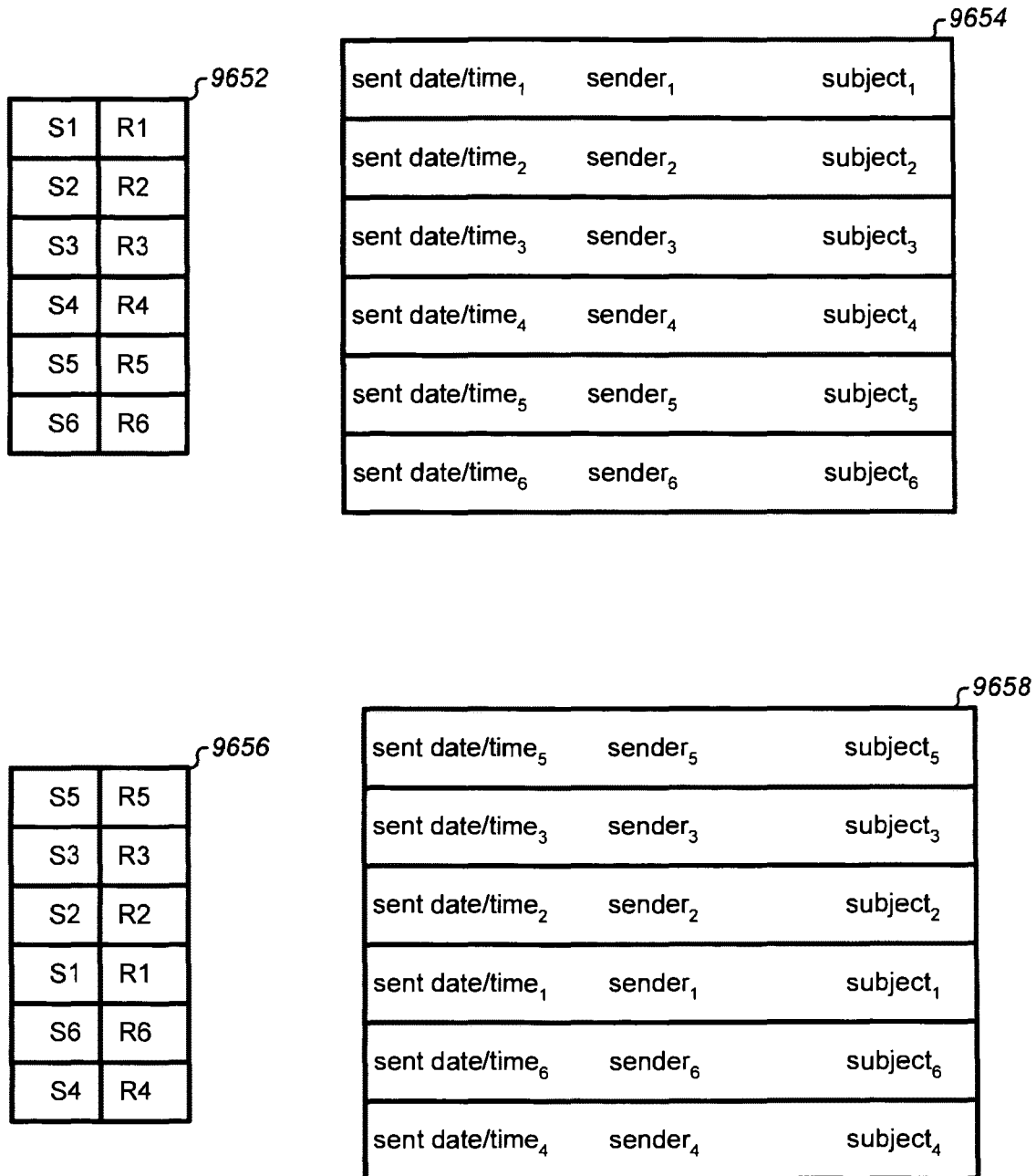


**Fig. 94B**

**Fig. 95A**

**Fig. 95B**

**Fig. 96A**

**Fig. 96B**

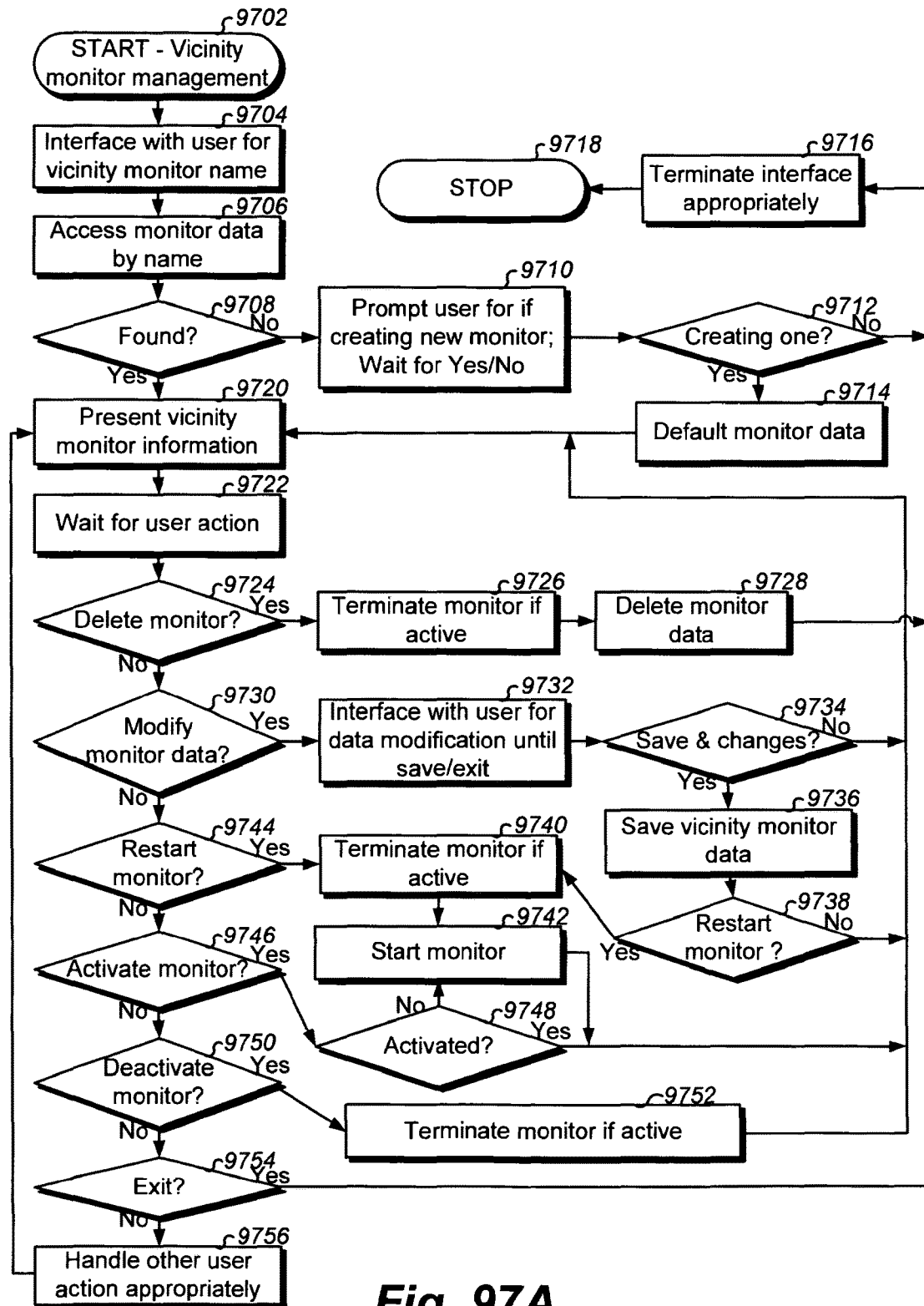


Fig. 97A

U.S. Patent

Nov. 19, 2019

Sheet 321 of 322

US 10,477,994 B2

William J. Johnson WJJ0802

321/322

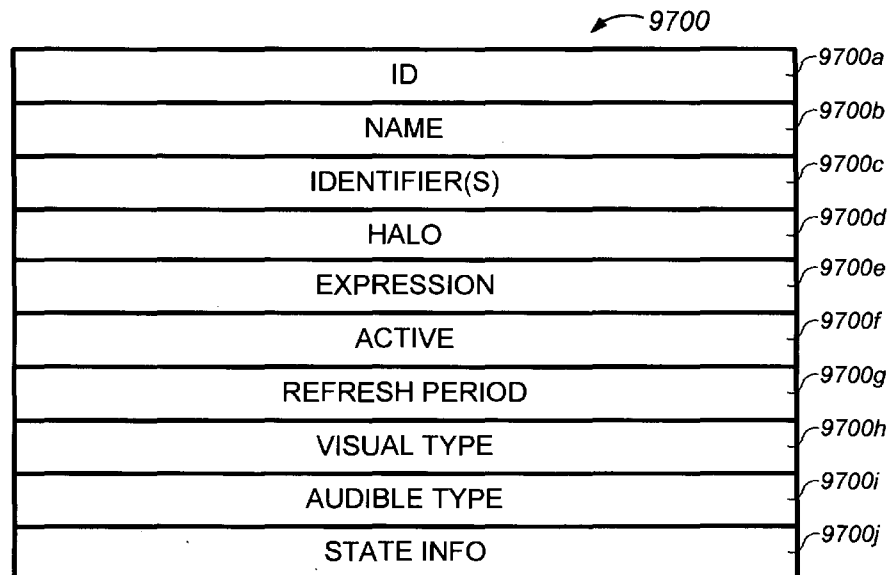
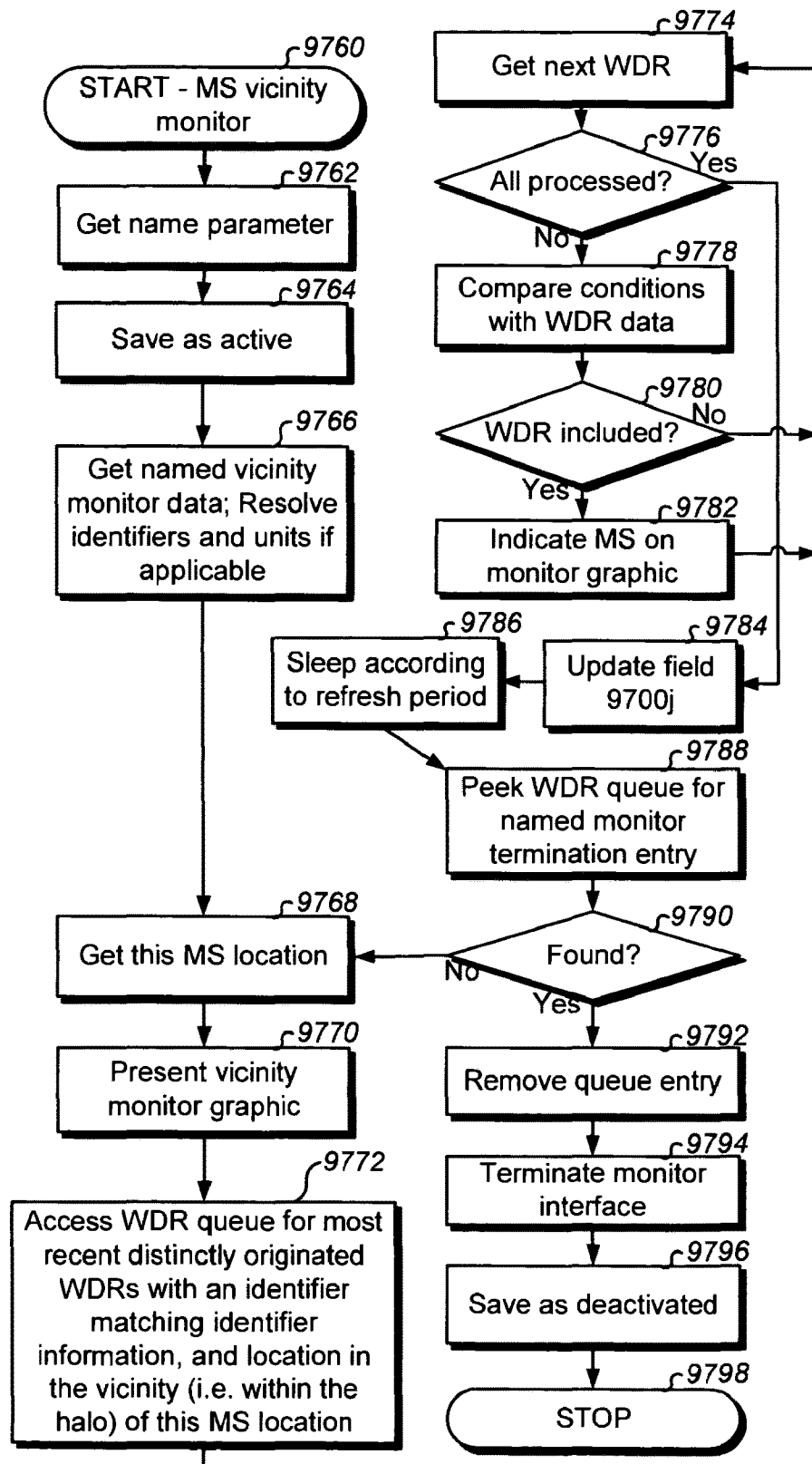


Diagram illustrating a table structure 9700, consisting of 10 rows, each labeled on the right side:

ID	9700a
NAME	9700b
IDENTIFIER(S)	9700c
HALO	9700d
EXPRESSION	9700e
ACTIVE	9700f
REFRESH PERIOD	9700g
VISUAL TYPE	9700h
AUDIBLE TYPE	9700i
STATE INFO	9700j

Fig. 97B

**Fig. 97C**

US 10,477,994 B2

1

SYSTEM AND METHOD FOR LOCATION BASED EXCHANGES OF DATA FACILITATING DISTRIBUTED LOCATIONAL APPLICATIONS

CROSS-REFERENCES TO RELATED APPLICATIONS

This application is a continuation of application Ser. No. 16/147,532 filed Sep. 28, 2018 and entitled "System and Method for Location Based Exchange Network" which is a continuation of application Ser. No. 15/218,039 (now U.S. Pat. No. 10,111,034 issued on Oct. 23, 2018) filed Jul. 24, 2016 and entitled "System and Method for Sound Wave Triggered Content Delivery" which is a continuation of application Ser. No. 14/752,945 (now U.S. Pat. No. 9,456,303 issued on Sep. 27, 2016) filed Jun. 28, 2015 and entitled "System and Method for Service Access Via Hopped Wireless Mobile Device(s)" which is a continuation of application Ser. No. 13/972,125 (now U.S. Pat. No. 9,078,095 issued on Jul. 7, 2015) filed Aug. 21, 2013 and entitled "System and Method for Location Based Inventory Management" which is a continuation of application Ser. No. 12/590,831 (now U.S. Pat. No. 8,634,796 issued on Jan. 21, 2014) filed Nov. 13, 2009 and entitled "System and Method for Location Based Exchanges of Data Facilitating Distributed Locational Applications" which is a continuation in part of application Ser. No. 12/287,064 (now U.S. Pat. No. 8,639,267 issued on Jan. 28, 2014) filed Oct. 3, 2008 and entitled "System and Method for Location Based Exchanges of Data Facilitating Distributed Locational Applications" which is a continuation in part of application Ser. No. 12/077,041 (now U.S. Pat. No. 8,600,341 issued on Dec. 3, 2013) filed Mar. 14, 2008 and entitled "System and Method for Location Based Exchanges of Data Facilitating Distributed Locational Applications". This application claims benefit of the earliest applicable application and contains an identical specification to Ser. No. 16/147,532 except for the title, abstract, and claims.

FIELD OF THE INVENTION

The present disclosure relates generally to location based services for mobile data processing systems, and more particularly to location based exchanges of data between distributed mobile data processing systems for locational applications. A common connected service is not required for location based functionality and features. Location based exchanges of data between distributed mobile data processing systems enable location based features and functionality in a peer to peer manner.

BACKGROUND OF THE INVENTION

The internet has exploded with new service offerings. Websites yahoo.com, google.com, ebay.com, amazon.com, and iTunes.com have demonstrated well the ability to provide valuable services to a large dispersed geographic audience through the internet (ebay, yahoo, google, amazon and iTunes (Apple) are trademarks of the respective companies). Thousands of different types of web services are available for many kinds of functionality. Advantages of having a service as the intermediary point between clients, users, and systems, and their associated services, includes centralized processing, centralized maintaining of data, for example to have an all knowing database for scope of services provided, having a supervisory point of control, providing an admin-

2

istrator with access to data maintained by users of the web service, and other advantages associated with centralized control. The advantages are analogous to those provided by the traditional mainframe computer to its clients wherein the mainframe owns all resources, data, processing, and centralized control for all users and systems (clients) that access its services. However, as computers declined in price and adequate processing power was brought to more distributed systems, such as Open Systems (i.e. Windows, UNIX, Linux, and Mac environments), the mainframe was no longer necessary for many of the daily computing tasks. In fact, adequate processing power is incorporated in highly mobile devices, various handheld mobile data processing systems, and other mobile data processing systems. Technology continues to drive improved processing power and data storage capabilities in less physical space of a device. Just as Open Systems took much of the load of computing off of mainframe computers, so to can mobile data processing systems offload tasks usually performed by connected web services. As mobile data processing systems are more capable, there is no need for a service to middleman interactions possible between them.

While a centralized service has its advantages, there are also disadvantages. A service becomes a clearinghouse for all web service transactions. Regardless of the number of threads of processing spread out over hardware and processor platforms, the web service itself can become a bottleneck causing poor performance for timely response, and can cause a large amount of data that must be kept for all connected users and/or systems. Even large web services mentioned above suffer from performance and maintenance overhead. A web service response will likely never be fast enough.

Additionally, archives must be kept to ensure recovery in the event of a disaster because the service houses all data for its operations. Archives also require storage, processing power, planning, and maintenance. A significantly large and costly data center is necessary to accommodate millions of users and/or systems to connect to the service. There is a tremendous amount of overhead in providing such a service. Data center processing power, data capacity, data transmission bandwidth and speed, infrastructure entities, and various performance considerations are quite costly. Costs include real estate required, utility bills for electricity and cooling, system maintenance, personnel to operate a successful business with service(s), etc. A method is needed to prevent large data center costs while eliminating performance issues for features sought. It is inevitable that as users are hungry for more features and functionality on their mobile data processing systems, processing will be moved closer to the device for optimal performance and infrastructure cost savings.

Service delivered location dependent content was disclosed in U.S. Pat. Nos. 6,456,234; 6,731,238; 7,187,997 (Johnson). Anonymous location based services was disclosed in U.S. PTO Publication 2006/0022048 (Johnson). The Johnson patents and published application operate as most web services do in that the clients connecting to the service benefit from the service by having some connectivity to the service. U.S. Publication 2006/0022048 (Johnson) could cause large numbers of users to inundate the service with device heartbeats and data to maintain, depending on the configurations made. While this may be of little concern to a company that has successfully deployed substantially large web service resources, it may be of great concern to other more frugal companies. A method is needed for

US 10,477,994 B2

3

enabling location dependent features and functionality without the burden of requiring a service.

Users are skeptical about their privacy as internet services proliferate. A service by its very nature typically holds information for a user maintained in a centralized service database. The user's preferences, credential information, permissions, customizations, billing information, surfing habits, and other conceivable user configurations and activity monitoring, can be housed by the service at the service. Company insiders, as well as outside attackers, may get access. Most people are concerned with preventing personal information of any type being kept in a centralized database which may potentially become compromised from a security standpoint. Location based services are of even more concern, in particular when the locations of the user are to be known to a centralized service. A method and system is needed for making users comfortable with knowing that their personal information is at less risk of being compromised.

A reasonable requirement is to push intelligence out to the mobile data processing systems themselves, for example, in knowing their own locations and perhaps the locations of other nearby mobile data processing systems. Mobile data processing systems can intelligently handle many of their own application requirements without depending on some remote service. Just as two people in a business organization should not need a manager to speak to each other, no two mobile data processing systems should require a service middleman for useful location dependent features and functionality. The knowing of its own location should not be the end of social interaction implementation local to the mobile data processing systems, but rather the starting place for a large number of useful distributed local applications that do not require a service.

Different users use different types of Mobile data processing Systems (MSs) which are also called mobile devices: laptops, tablet computers, Personal Computers (PCs), Personal Digital Assistants (PDAs), cell phones, automobile dashboard mounted data processing systems, shopping cart mounted data processing systems, mobile vehicle or apparatus mounted data processing systems, Personal Navigational Devices (PNDs), iPhones (iPhone is a trademark of Apple, Inc.), various handheld mobile data processing systems, etc. MSs move freely in the environment, and are unpredictably moveable (i.e. can be moved anywhere, anytime). Many of these Mobile data processing Systems (MSs) do not have capability of being automatically located, or are not using a service for being automatically located. Conventional methods use directly relative stationary references such as satellites, antennas, etc. to locate MSs. Stationary references are expensive to deploy, and risk obsolescence as new technologies are introduced to the marketplace.

Stationary references have finite scope of support for locating MSs.

While the United States E911 mandate for cellular devices documents requirements for automatic location of a Mobile data processing System (MS) such as a cell phone, the mandate does not necessarily promote real time location and tracking of the MSs, nor does it define architecture for exploiting Location Based Services (LBS). We are in an era where Location Based Services (LBS), and location dependent features and functionality, are among the most promising technologies in the world. Automatic locating of every Mobile data processing System (MS) is an evolutionary trend. A method is needed to shorten the length of time for automatically locating every MS. Such a goal can be costly

4

using prior art technologies such as GPS (Global Positioning System), radio wave triangulation, coming within range to a known located sensor, or the like. Complex system infrastructure, or added hardware costs to the MSs themselves, make such ventures costly and time constrained by schedules and costs involved in engineering, construction, and deployment.

A method is needed for enabling users to get location dependent features and functionality through having their mobile locations known, regardless of whether or not their MS is equipped for being located. Also, new and modern location dependent features and functionality can be provided to a MS unencumbered by a connected service.

BRIEF SUMMARY OF THE INVENTION

LBS (Location Based Services) is a term which has gained in popularity over the years as MSs incorporate various location capability. The word "Services" in that terminology plays a major role in location based features and functionality involving interaction between two or more users. This disclosure introduces a new terminology, system, and method referred to as Location Based eXchanges (LBX). LBX is an acronym used interchangeably/contextually throughout this disclosure for the singular term "Location Based Exchange" and for the plural term "Location Based Exchanges", much the same way LBS is used interchangeably/contextually for the single term "Location Based Service" and for the plural term "Location Based Services". LBX describes leveraging the distributed nature of connectivity between MSs in lieu of leveraging a common centralized service nature of connectivity between MSs. The line can become blurred between LBS and LBX since the same or similar features and functionality are provided, and in some cases strengths from both may be used. The underlying architectural shift differentiates LBX from LBS for depending less on centralized services, and more on distributed interactions between MSs. LBX provide server-free and server-less location dependent features and functionality.

Disclosed are many different aspects to LBX, starting with the foundation requirement for each participating MS to know, at some point in time, their own whereabouts. LBX is enabled when an MS knows its own whereabouts. It is therefore a goal to first make as many MSs know their own whereabouts as possible. When two or more MSs know their own whereabouts, LBX enables distributed locational applications whereby a server is not required to middleman social interactions between the MSs. The MSs interact as peers. LBX disclosed include purely peer to peer interactions, peer to peer interactions for routing services, peer to peer interactions for delivering distributed services, and peer to peer interactions for location dependent features and functionality (e.g. a first mobile data processing system sends directly (e.g. wirelessly) to a second mobile data processing system without using an intervening data processing system). One embodiment of an LBX enabled MS is referred to as an lbxPhone™.

It is an advantage herein to have no centralized service governing location based features and functionality among MSs. Avoiding a centralized service prevents performance issues, infrastructure costs, and solves many of the issues described above. No centralized service also prevents a user's information from being kept in one accessible place. LBS contain centralized data that is personal in nature to its users. This is a security concern. Having information for all users in one place increases the likelihood that a disaster to

US 10,477,994 B2

5

the data will affect more than a single user. LBX spreads data out across participating systems so that a disaster affecting one user does not affect any other user.

It is an advantage herein for enabling useful distributed applications without the necessity of having a service, and without the necessity of users and/or systems registering with a service. MSs interact as peers in preferred embodiments, rather than as clients to a common service (e.g. internet connected web service).

It is an advantage herein for locating as many MSs as possible in a wireless network, and without additional deployment costs on the MSs or the network. Conventional locating capability includes GPS (Global Positioning System) using stationary orbiting satellites, improved forms of GPS, for example AGPS (Adjusted GPS) and DGPS (Differential GPS) using stationary located ground stations, wireless communications to stationary located cell tower base stations, TDOA (Time Difference of Arrival) or AOA (Angle of Arrival) triangulation using stationary located antennas, presence detection in vicinity of a stationary located antenna, presence detection at a wired connectivity stationary network location, or other conventional locating systems and methods. Mobile data processing systems, referred to as Indirectly Located Mobile data processing systems (ILMs), are automatically located using automatically detected locations of Directly Located Mobile data processing systems (DLMs) and/or automatically detected locations of other ILMs. ILMs are provided with the ability to participate in the same LBS, or LBX, as a DLM (Directly Located Mobile data processing system). DLMs are located using conventional locating capability mentioned above. DLMs provide reference locations for automatically locating ILMs, regardless of where any one is currently located. DLMs and ILMs can be highly mobile, for example when in use by a user. There are a variety of novel methods for automatically locating ILMs, for example triangulating an ILM (Indirectly Located Mobile data processing system) location using a plurality of DLMs, detecting the ILM being within the vicinity of at least one DLM, triangulating an ILM location using a plurality of other ILMs, detecting the ILM being within the vicinity of at least one other ILM, triangulating an ILM location using a mixed set of DLM(s) and ILM(s), determining the ILM location from heterogeneously located DLMs and/or ILMs, and other novel methods.

MSs are automatically located without using direct conventional means for being automatically located. The conventional locating capability (i.e. conventional locating methods) described above is also referred to as direct methods. Conventional methods are direct methods, but not all direct methods are conventional. There are new direct techniques disclosed below. Provided herein is an architecture, as well as systems and methods, for immediately bringing automatic location detection to every MS in the world, regardless of whether that MS is equipped for being directly located. MSs without capability of being directly located are located by leveraging the automatically detected locations of MSs that are directly located. This is referred to as being indirectly located. An MS which is directly located is hereinafter referred to as a Directly Located Mobile data processing system (DLM). For a plural acronym, MSs which are directly located are hereinafter referred to as Directly Located Mobile data processing systems (DLMs). MSs without capability of being directly located are located using the automatically detected locations of MSs that have already been located. An MS which is indirectly located is hereinafter referred to as an Indirectly Located Mobile data

6

processing system (ILM). For a plural acronym, MSs which are indirectly located are hereinafter referred to as Indirectly Located Mobile data processing systems (ILMs). A DLM can be located in the following ways:

- A) New triangulated wave forms;
- B) Missing Part Triangulation (MPT) as disclosed below;
- C) Heterogeneous direct locating methods;
- D) Assisted Direct Location Technology (ADLT) using a combination of direct and indirect methods;
- E) Manually specified; and/or
- F) Any combinations of A) through E);

DLMs provide reference locations for automatically locating ILMs, regardless of where the DLMs are currently located. It is preferable to assure an accurate location of every DLM, or at least provide a confidence value of the accuracy. A confidence value of the accuracy is used by relative ILMs to determine which are the best set (e.g. which are of highest priority for use to determine ILM whereabouts) of relative DLMs (and/or ILMs) to use for automatically determining the location of the ILM.

In one example, the mobile locations of several MSs are automatically detected using their local GPS chips. Each is referred to as a DLM. The mobile location of a non-locatable MS is triangulated using radio waves between it and three (3) of the GPS equipped DLMs. The MS becomes an ILM upon having its location determined relative the DLMs. ILMs are automatically located using DLMs, or other already located ILMs. An ILM can be located in the following ways:

- G) Triangulating an ILM location using a plurality of DLMs with wave forms of any variety (e.g. AOA, TDOA, MPT (a heterogeneous location method));
- H) Detecting the ILM being within the reasonably close vicinity of at least one DLM;
- I) Triangulating an ILM location using a plurality of other ILMs with wave forms of any variety;
- J) Detecting the ILM being within the reasonable close vicinity of at least one other ILM;
- K) Triangulating an ILM location using a mixed set of DLM(s) and ILM(s) with wave forms of any variety (referred to as ADLT);
- L) Determining the ILM location from heterogeneously located DLMs and/or ILMs (i.e. heterogeneously located, as used here, implies having been located relative different location methodologies);
- M) A) through F) Above; and/or
- N) Any combinations of A) through M).

Locating functionality may leverage GPS functionality, including but not limited to GPS, AGPS (Adjusted GPS), DGPS, (Differential GPS), or any improved GPS embodiment to achieve higher accuracy using known locations, for example ground based reference locations. The Nextel GPS enabled iSeries cell phones provide excellent examples for use as DLMs (Nextel is a trademark of Sprint/Nextel). Locating functionality may incorporate triangulated locating of the MS, for example using a class of Radio Frequency (RF) wave spectrum (cellular, WiFi (some WiFi embodiments referred to as WiMax), bluetooth, etc), and may use measurements from different wave spectrums for a single location determination (depends on communications interface(s) available). A MS may have its whereabouts determined using a plurality of wave spectrum classes available to it (cellular, WiFi, bluetooth, etc). The term "WiFi" used throughout this disclosure also refers to the industry term "WiMax". Locating functionality may include in-range proximity detection for detecting the presence of the MS. Wave forms for triangulated locating also include

US 10,477,994 B2

7

microwaves, infrared wave spectrum relative infrared sensors, visible light wave spectrum relative light visible light wave sensors, ultraviolet wave spectrum relative ultraviolet wave sensors, X-ray wave spectrum relative X-ray wave sensors, gamma ray wave spectrum relative gamma ray wave sensors, and longwave spectrum (below AM) relative longwave sensors. While there are certainly more common methods for automatically locating a MS (e.g. radio wave triangulation, GPS, in range proximity detection), those skilled in the art recognize there are methods for different wave spectrums being detected, measured, and used for carrying information between data processing systems.

Kubler et al (U.S. PTO publications 2004/0264442, 2004/0246940, 2004/0228330, 2004/0151151) disclosed methods for detecting presence of mobile entities as they come within range of a sensor. In Kubler et al, accuracy of the location of the detected MS is not well known, so an estimated area of the whereabouts of the MS is enough to accomplish intended functionality, for example in warehouse installations. A confidence value of this disclosure associated with Kubler et al tends to be low (i.e. not confident), with lower values for long range sensors and higher values for short range sensors.

GPS and the abundance of methods for improving GPS accuracy has led to many successful systems for located MSs with high accuracy. Triangulation provides high accuracies for locating MSs. A confidence value of this disclosure associated with GPS and triangulating location methods tends to be high (i.e. confident). It is preferred that DLMs use the highest possible accuracy method available so that relative ILMs are well located. Not all DLMs need to use the same location methods. An ILM can be located relative DLMs, or other ILMs, that each has different locating methodologies utilized.

Another advantage herein is to generically locate MSs using varieties and combinations of different technologies. MSs can be automatically located using direct conventional methods for accuracy to base on the locating of other MSs. MSs can be automatically located using indirect methods. Further, it is an advantage to indirectly locate a MS relative heterogeneously located MSs. For example, one DLM may be automatically located using GPS. Another DLM may be automatically located using cell tower triangulation. A third DLM may be automatically located using within range proximity. An ILM can be automatically located at a single location, or different locations over time, relative these three differently located DLMs. The automatically detected location of the ILM may be determined using a form of triangulation relative the three DLMs just discussed, even though each DLM had a different direct location method used. In a preferred embodiment, industry standard IEEE 802.11 WiFi is used to locate (triangulate) an ILM relative a plurality of DLMs (e.g. TDOA in one embodiment). This standard is prolific among more compute trended MSs. Any of the family of 802.11 wave forms such as 802.11a, 802.11b, 802.11g, or any other similar class of wave spectrum can be used, and the same spectrum need not be used between a single ILM and multiple DLMs. 802.x used herein generally refers to the many 802.whatever variations.

Another advantage herein is to make use of existing marketplace communications hardware, communications software interfaces, and communications methods and location methods where possible to accomplish locating a MS relative one or more other MSs. While 802.x is widespread for WiFi communications, other RF wave forms can be used (e.g. cell phone to cell tower communications). In fact, any wave spectrum for carrying data applies herein. Of course,

8

any protocol(s) may be involved in embodiments of the disclosures (e.g. TDMA, CDMA, H.323, SIP, 2G, 3G, ip phone, digital, analog, spectrum frequency, etc).

Still another advantage is for support of heterogeneous locatable devices. Different people like different types of devices as described above. Complete automation of locating functionality can be provided to a device through local automatic location detection means, or by automatic location detection means remote to the device. Also, an ILM can be located relative a laptop, a cell phone, and a PDA (i.e. different device types).

Yet another advantage is to prevent the unnecessary storing of large amounts of positioning data for a network of MSs. Keeping positioning data for knowing the whereabouts of all devices can be expensive in terms of storage, infrastructure, performance, backup, and disaster recovery. A preferred embodiment simply uses a distributed approach to determining locations of MSs without the overhead of an all-knowing database maintained somewhere. Positions of MSs can be determined "on the fly" without storing information in a master database. However, there are embodiments for storing a master database, or a subset thereof, to configurable storage destinations, when it makes sense. A subset can be stored at a MS.

Another advantage includes making use of existing location equipped MSs to expand the network of locatable devices by locating non-equipped MSs relative the location of equipped MSs. MSs themselves help increase dimensions of the locatable network of MSs. The locatable network of MSs is referred to as an LN-Expanse (i.e. Location-Network Expanse). An LN-Expanse dynamically grows and shrinks based on where MSs are located at a particular time. For example, as users travel with their personal MSs, the personal MSs themselves define the LN-Expanse since the personal MSs are used to locate other MSs. An ILM simply needs location awareness relative located MSs (DLMs and/or ILMs).

Yet another advantage is a MS interchangeably taking on the role of a DLM or ILM as it travels. MSs are chameleons in this regard, in response to location technologies that happen to be available. A MS may be equipped for DLM capability, but may be in a location at some time where the capability is inoperable. In these situations the DLM takes on the role of an ILM. When the MS again enters a location where it can be a DLM, it automatically takes on the role of the DLM. This is very important, in particular for emergency situations. A hiker has a serious accident in the mountains which prevents GPS equipped DLM capability from working. Fortunately, the MS automatically takes on the role of an ILM and is located within the vicinity of neighboring (nearby) MSs. This allows the hiker to communicate his location, operate useful locational application functions and features at his MS, and enable emergency help that can find him.

It is a further advantage that MS locations be triangulated using any wave forms (e.g. RF, microwaves, infrared, visible light, ultraviolet, X-ray, gamma ray). X-ray and gamma ray applications are special in that such waves are harmful to humans in short periods of times, and such applications should be well warranted to use such wave forms. In some medical embodiments, micro-machines may be deployed within a human body. Such micro-machines can be equipped as MSs. Wave spectrums available at the time of deployment can be used by the MSs for determining exact positions when traveling through a body.

It is another advantage to use TDOA (Time Difference Of Arrival), AOA (Angle Of Arrival), and Missing Part Triangulation.

US 10,477,994 B2

9

gulation (MPT) when locating a MS. TDOA uses time information to determine locations, for example for distances of sides of a triangle. AOA uses angles of arrival to antennas to geometrically assess where a MS is located by intersecting lines drawn from the antennas with detected angles. MPT is disclosed herein as using combinations of AOA and TDOA to determine a location. Exclusively using all AOA or exclusively using all TDOA is not necessary. MPT can be a direct method for locating MSs.

Yet another advantage is to locate MSs using Assisted Direct Location Technology (ADLT). ADLT is disclosed herein as using direct (conventional) location capability together with indirect location capability to confidently determine the location of a MS.

Still another advantage is to permit manual specification for identifying the location of a MS (a DLM). The manual location can then in turn be used to facilitate locating other MSs. A user interface may be used for specification of a DLM location. The user interface can be local, or remote, to the DLM. Various manual specification methods are disclosed. Manual specification is preferably used with less mobile MSs, or existing MSs such as those that use dodgeball.com (trademark of Google). The confidence value depends on how the location is specified, whether or not it was validated, and how it changes when the MS moves after being manually set. Manual specification should have limited scope in an LN-expanse unless inaccuracies can be avoided.

Another advantage herein is locating a MS using any of the methodologies above, any combinations of the methodologies above, and any combinations of direct and/or indirect location methods described.

Another advantage is providing synergy between different locating technologies for smooth operations as an MS travels. There are large numbers of methods and combinations of those methods for keeping an MS informed of its whereabouts. Keeping an MS informed of its whereabouts in a timely manner is critical in ensuring LBX operate optimally, and for ensuring nearby MSs without certain locating technologies can in turn be located.

It is another advantage for locating an MS with multiple location technologies during its travels, and in using the best of breed data from multiple location technologies to infer a MS location confidently. Confidence values are associated with reference location information to ensure an MS using the location information can assess accuracy. A DLM is usually an "affirmifier". An affirmifier is an MS with its whereabouts information having high confidence of accuracy and can serve as a reference for other MSs. An ILM can also be an affirmifier provided there is high confidence that the ILM location is known. An MS (e.g. ILM) may be a "pacifier". A pacifier is an MS having location information for its whereabouts with a low confidence for accuracy. While it can serve as a reference to other ILMs, it can only do so by contributing a low confidence of accuracy.

It is another advantage for providing user customization of confidence values based on the user's experience. A MS user may completely rely on the MS system settings for setting confidence values, or may "tweak" location technology confidence values to accommodate experiences with particular location technologies that have been encountered during travels.

It is an advantage to synergistically make use of the large number of locating technologies available to prevent one particular type of technology to dominate others while using the best features of each to assess accurate mobile locations of MSs.

10

A further advantage is to leverage a data processing system with capability of being located for co-locating another data processing system without any capability of being located. For example, a driver owns an older model automobile, has a useful second data processing system in the automobile without means for being automatically located. The driver also own a cell phone, called a first data processing system, which does have means for being automatically located. The location of the first data processing system can be shared with the second data processing system for locating the second data processing system. Further still, the second data processing system without means for being automatically located is located relative a first set (plurality) of data processing systems which are not at the same location as the second data processing system. So, data processing systems are automatically located relative at least one other data processing which can be automatically located.

Another advantage is a LBX enabled MS includes a service informant component for keeping a supervisory service informed. This prevents an MS from operating in total isolation, and prevents an MS from operating in isolation with those MSs that are within its vicinity (e.g. within maximum range **1306**) at some point in time, but to also participate when the same MSs are great distances from each other. There are LBX which would fit well into an LBS model, but a preferred embodiment chooses to use the LBX model. For example, multiple MS users are seeking to carpool to and from a common destination. The service informant component can perform timely updates to a supervisory service for route comparisons between MSs, even though periods of information are maintained only at the MSs. For example, users find out that they go to the same church with similar schedules, or coworkers find out they live nearby and have identical work schedules. The service informant component can keep a service informed of MS whereabouts to facilitate novel LBX applications. The service informant can also be configured for: communicating directly to another MS, communicating to a data processing system through a propagate-able service, invoking a "plug-in" home grown interface, alerting the MS user with a specified alert, or invoking an atomic command used by charter processing.

It is a further advantage in leveraging the vast amount of MS WiFi/WiMax deployment underway in the United States. More widespread WiFi/WiMax availability enhances the ability for well performing peer to peer types of features and functionality disclosed.

It is a further advantage to prevent unnecessary established connections from interfering with successfully triangulating a MS position. As the MS roams and encounters various wave spectrum signals, that is all that is required for determining the MS location. Broadcast signaling contains the necessary location information for automatically locating the MS.

Yet another advantage is to leverage Network Time Protocol (NTP) for eliminating bidirectional communications in determining Time of Arrival (TOA) and TDOA (Time Difference Of Arrival) measurements (TDOA as used in the disclosure generally refers to both TOA and TDOA). NTP enables a single unidirectional transmission of data to carry all that is necessary in determining TDOA, provided the sending data processing system and the receiving data processing system are NTP synchronized to an adequate granulation of time.

A further advantage is for making available to remote peer MSs certain MS operating system resources such as

US 10,477,994 B2

11

memory, storage, semaphores, application data, or the like, according to permissions. A single MS can access and use operating system resources of another MS, for example in charter processing. Also, semaphore controlled synchronization of processing can be achieved over a network, or plurality, of peer MSs without a common server to synchronize the processing.

It is an advantage of this disclosure to provide a competing superior alternative to server based mobile technologies such as that of U.S. Pat. Nos. 6,456,234; 6,731,238; 7,187,997; and U.S. PTO Publication 2006/0022048 (Johnson). It is also an advantage to leverage both LBX technology and LBS technology in the same MS in order to improve the user experience. The different technologies can be used to complement each other in certain embodiments.

A further advantage herein is to leverage existing "usual communications" data transmissions for carrying new data that is ignored by existing MS processing, but observed by new MS processing, for carrying out processing maximizing location functions and features across a large geography. Alternatively, new data can be transmitted between systems for the same functionality.

It is an advantage herein in providing peer to peer service propagation. ILMs are provided with the ability to participate in the same Location Based Services (LBS) or other services as DLM(s) in the vicinity. An MS may have access to services which are unavailable to other MSs. Any MS can share its accessible services for being accessible to any other MS, preferably in accordance with permissions. For example, an MS without internet access can get internet access via an MS in the vicinity with internet access. In a preferred embodiment, permissions are maintained in a peer to peer manner prior to lookup for proper service sharing. In another embodiment, permissions are specified and used at the time of granting access to the shared services. Once granted for sharing, services can be used in a mode as if the sharing user is using the services, or in a mode as if the user accepting the share is a new user to the service. Routing paths are dynamically reconfigured and transparently used as MSs travel. Hop counts dynamically change to strive for a minimal number of hops for an MS getting access to a desirable service. Route communications depend on where the MS needing the service is located relative a minimal number of hops through other MSs to get to the service. Services can be propagated from DLMs to DLMs, DLMs to ILMs, ILMs to DLMs, or ILMs to ILMs.

Services otherwise unavailable to a first MS (or MS user) in the LN-Expanse become available through another MS which does have access to the service. A plurality of MSs may facilitate the connection (e.g. hops) from the first MS to the last MS which publishes the service and has access to the service. MSs can access needed services through MSs in the vicinity when necessary. A service directory is shared and propagated between MSs so that the superset of services in a LN-Expanse are made available to any one MS in the LN-Expanse regardless of current MS conditions, whereabouts, capability, or an inability to connect to a desired service. A service route is minimized for best performance even with highly mobile MSs by minimizing a number of hops between MSs to reach a service.

It is another advantage herein for providing peer to peer permissions, authentication, and access control. A service is not necessary for maintaining credentials and permissions between MSs. Permissions are maintained locally to a MS. In a centralized services model, a database can become massive in size when searching for needed permissions. Permission searching and validation of U.S. PTO Publica-

12

tion 2006/0022048 (Johnson) was costly in terms of database size and performance. There was overhead in maintaining who owned the permission configuration for every permission granted. Maintaining permissions locally, as described below, reduces the amount of data to represent the permission because the owner is understood to be the personal user of the MS. Additionally, permission searching is very fast because the MS only has to search its local data for permissions that apply to only its MS.

Yet another advantage is to provide a nearby, or nearness, status using a peer to peer system and method, rather than intelligence maintained in a centralized database for all participating MSs. There is lots of overhead in maintaining a large database containing locations of all known MSs. This disclosure removes such overhead through using nearby detection means of one MS when in the vicinity of another MS. There are varieties of controls for governing how to generate the nearby status. In one aspect, a MS automatically calls the nearby MS thereby automatically connecting the parties to a conversation without user interaction to initiate the call. In another aspect, locally maintained configurations govern functionality when MSs are newly nearby, or are newly departing being nearby. Nearby status, alerts, and queries are achieved in a LBX manner.

It is yet another advantage for automatic call forwarding, call handling, and call processing based on the whereabouts of a MS, or whereabouts of a MS relative other MSs. The nearness condition of one MS to another MS can also affect the automatic call forwarding functionality.

Yet another advantage herein is for peer to peer content delivery and local MS configuration of that content. Users need no connectivity to a service. Users make local configurations to enjoy location based content delivery to other MSs. Content is delivered under a variety of circumstances for a variety of configurable reasons. Content maintained local to an MS is delivered asynchronously to other MSs for nearby alerts, arrival or departure to and from geofenced areas, and other predicated conditions of nearby MSs. While it may appear there are LBS made available to users of MSs, there are in fact LBX being made available to those users.

Another advantage herein is a LBX enabled MS can operate in a peer to peer manner to data processing systems which control environmental conditions. For example, automobile equipped (or driver kept) MSs encounter an intersection having a traffic light. Interactions between the MSs at the intersection and a data processing system in the vicinity for controlling the traffic light can automatically override light color changing for optimal traffic flow. In another embodiment, a parking lot search by a user with an MS is facilitated as he enters the parking lot, and in accordance with parking spaces currently occupied. In general, other nearby data processing systems can have their control logic processed for a user's preferences (as defined in the MS), a group of nearby user's preferences, and/or situational locations (see U.S. Pat. Nos. 6,456,234; 6,731,238; 7,187,997 (Johnson) for "situational location" terminology) of nearby MSs.

Another advantage herein is an MS maintains history of hotspot locations detected for providing graphical indication of hotspot whereabouts. This information can be used by the MS user in guiding where a user should travel in the future for access to services at the hotspot. Hotspot growth prevents a database in being timely configured with new locations. The MS can learn where hotspots are located, as relevant to the particular MS. The hotspot information is instantly available to the MS.

US 10,477,994 B2

13

A further advantage is for peer to peer proximity detection for identifying a peer service target within the MS vicinity. A peer service target can be acted upon by an MS within range, using an application at the MS. The complementary whereabouts of the peer service target and MS automatically notify the user of service availability. The user can then use the MS application for making a payment, or for performing an account transfer, account deposit, account deduction, or any other transaction associated with the peer service target.

Yet another advantage is for a MS to provide new self management capability such as automatically marking photographs taken with location information, a date/time stamp, and who was with the person taking the picture.

Yet another advantage is being alerted to nearby people needing assistance and nearby fire engines or police cars that need access to roads.

A further advantage is providing a MS platform for which new LBX features and functionality can be brought quickly to the marketplace. The platform caters to a full spectrum of users including highly technical software developers, novice users, and users between those ranges. A rich programming environment is provided wherein whereabouts (WDR) information interchanged with other MSs in the vicinity causes triggering of privileged actions configured by users. The programming environment can be embedded in, or “plugged into”, an existing software development environment, or provided on its own. A syntax may be specified with source code statements, XML, SQL database definitions, a datastream, or any other derivative of a well defined BNF grammar. A user friendly configuration environment is provided wherein whereabouts information interchanged with other MSs in the vicinity causes triggering of privileged actions configured by users. The platform is an event based environment wherein WDRs containing certain configured sought information are recognized at strategic processing paths for causing novel processing of actions. Events can be defined with complex expressions, and actions can be defined using homegrown executables, APIs, scripts, applications, a set of commands provided with the LBX platform, or any other executable processing. The LBX platform includes a variety of embodiments for charter and permission definitions including an internalized programmatic form, a SQL database form, a data record form, a datastream form, and a well defined BNF grammar for deriving other useful implementations (e.g. lex and yacc).

It is an advantage for permissions and/or charters to be configured in anticipation of every possible future travel, situation, environment, application, or condition of a MS (or MS user), or a plurality of related (by permissions and charters) MSs (or MS users). It is powerful in how permissions and charters configured in advance of anticipated events reveal novel unpredictably timed automated actions and application behavior for novel uses.

It is another advantage to support a countless number of privileges that can be configured, managed, and processed in a peer to peer manner between MSs. Any peer to peer feature or set of functionality can have a privilege associated to it for being granted from one user to another. It is also an advantage for providing a variety of embodiments for how to manage and maintain privileges in a network of MSs.

It is another advantage to support a complete set of options for charters that can be configured, managed, and processed in a peer to peer manner between MSs. Charters can become effective under a comprehensive set of conditions, expressions, terms, and operators. It is also an advantage for providing a variety of embodiments for how to manage and maintain charters in a network of MSs. Charters

14

themselves can be self modifying for changing permissions or charters “on the fly” (i.e. during charter processing).

It is a further advantage for providing multithreaded communications of permission and charter information and transactions between MSs for well performing peer to peer interactions. Any signal spectrum for carrying out transmission and reception is candidate, depending on the variety of MS. In fact, different signaling wave spectrums, types, and protocols may be used in interoperating communications, or even for a single transaction, between MSs.

It is yet another advantage for increasing the range of the LN-expanse from a wireless vicinity to potentially infinite vicinity through other data processing (e.g. routing) equipment. While wireless proximity is used for governing automatic location determination, whereabouts information may be communicated between MSs great distances from each other provided there are privileges and/or charters in place making such whereabouts information relevant for the MS. Whereabouts information of others will not be maintained unless there are privileges in place to maintain it. Whereabouts information may not be shared with others if there have been no privileges granted to a potential receiving MS. Privileges can provide relevance to what whereabouts (WDR) information is of use, or should be processed, maintained, or acted upon.

Another advantage is to provide a MS which can be user configured for any desired behavior based on location, whereabouts, and “in the vicinity” conditions for the MS and/or its peer MSs during travels. A user has infinite control over providing a processing “character” for the MS. Also, various MS applications are generically supported with integrated locational based features and functionality. Charters may be used to automatically perform: MS configuration and system variable setting, clip-board and paste operations, MS input and output control, automatic communications with other MSs or data processing systems, enabling/disabling a feature or service, and many other features.

Another advantage is for using a convenient user interface such as map navigation for generating a map term such as a point, point and radius, or set of points defining area(s) on a map which is conveniently referenced in a charter configuration and later processed for replacement. For example, a user makes selection(s) on a map, and location information is automatically generated for the selection(s). The user can assign a convenient name to the location information without knowing details of the location information itself. The user can then reference the name for completely specifying the associated location information details. Also, the user may use WDR search criteria for determining a map term, the WDR found being one originated from the MS of map term creation or that of a peer MS. Recent whereabouts of a WDR found (e.g. from queue 22), or past whereabouts of a WDR found (e.g. history 30) may be used. Queue 22 may be viewed as maintaining a short term history, while history 30 may be viewed as maintaining a longer term history. Specifying locations in charter configurations can be tedious. Map terms provide the user with a simple user interface method to specify locations, and for hiding complexities of how the location was determined and generated for charter use. In some embodiments, map terms are used in broader scope by permitting any substitution where referenced. In some embodiments, map terms are used in broader scope by permitting “special terms” to be automatically created by a user by simply selecting a MS on a map.

It is an advantage for a convenient “charters starters” user interface for browsing, enabling, disabling, and maintaining

US 10,477,994 B2

15

charters depending on application, categories, or useable/clone-able snippets of the charters. For example, a MS may come prepackaged with many charters which have been organized and marked for particular applications and categories. The user can search, find, manage and enable/disable a set of charters based on their application or category, and can clone charter subsets for creating new charters. A MS user may manage his own charters, or charters of privilege granting others, using the charters starters interfaces. The user is also able to search, find, manage and enable/disable a set of charters based on any criteria found in the charter definitions themselves. A knowledgeable or authorized user may organize charters as he sees fit, for example to assign charters to categories and applications. The charter starters user interface organizes charters in easily identifiable groups (e.g. folders, categories, applications, etc) and provides simplicity for enabling, disabling and organizing any desired sets of complex charter configurations.

It is an advantage in providing application term triggered processing to the LBX platform described, and for all users and skill sets thereof. A rich programming environment and user friendly configuration environment is provided wherein application data which becomes modified causes triggering of privileged actions configured by users. The programming environment can be embedded in, or "plugged into", an existing software development environment, or provided on its own. A syntax may be specified with source code statements, XML, SQL database definitions, a datastream, or any other derivative of the disclosed BNF grammar. The platform is an event based environment wherein events of modifying application data containing configured sought values/information are recognized for triggering processing of actions. Events can be defined with complex expressions, and actions can be defined using homegrown executables, APIs, scripts, applications, a set of commands provided with the LBX platform, or any other executable processing. The LBX platform includes a variety of embodiments as described.

Another advantage is providing a comprehensive palette of paste commands for pasting LBX data into data entry fields, snapshot images, or one or more video stream frames. Data can be accessed and used for pasting from: queue 22; history 30; statistics 14; service directory 16; atomic terms; map terms; WDRTerm data; AppTerm data; any term or construct of the LBX BNF grammar; data describing current, past or future LBX data; averages of MS or LBX data; data derived from MSs in the vicinity (e.g. nearby); and data sensed, received, sent, processed, analyzed, or predicted at the MS. Data being pasted may be converted prior to the paste as a user requests. The user may adjust the paste data appearance (font, size, color, or any other appearance characteristic) prior to finalizing the paste action.

Yet another advantage is providing "plug-in" application support so that an application can be integrated conveniently into the LBX architecture and framework through Prefix Registry Records 5300. Application data and executable interfaces are "plugged in". Application data is made accessible to charter processing for conditional and configurable event based charter processing. Various "plug-in" systems and methods are described. The LBX platform is designed to integrate well with MS applications of all varieties for a cohesive architecture.

Another advantage is for tightly coupling/integrating LBX processing configuration and processing into a programming environment for a WPL in context of a rich PPL. LBX processing can be a "plug-in" to PPLs, or may be

16

integrated into the PPL syntax for a rich WPL. There are a variety of systems and methods described for a comprehensive LBX platform.

It is an advantage for facilitating the creation of charters that make sense in context of a particular MS application by automating suggestions. Special terms and atomic operands are determined for an application context, and candidate charters and/or portions thereof are presented for use to the user based on being derived from the special terms and atomic operands determined for the application context. A user's effort in creating charters for a particular application context is minimized with ready-made charters or charter portions that are automatically determined to be relevant for the particular application context. Upon being presented with suggestions, the user can select, or select and "tweak" to a desired charter configuration. The user can also configure privileges that are in context of the application or the charters selected.

It is an advantage for automatically comparing MS data profile information for matches for triggering conditional actions of charters. Users can configure data which is beacons to other MSs and then compared for matches for automated charter processing. MSs are automated with social interaction to other MSs so that MS users are alerted of MS users of interest in the vicinity for a variety of applications.

It is an advantage for transmitting application data fields to peer MSs in the vicinity, receiving application data fields from peer MSs in the vicinity, transmitting application data fields to data processing systems in the vicinity in a peer to peer manner, and receiving application data fields from data processing systems in the vicinity in a peer to peer manner for interoperability of a diverse set of applications and automated triggered processing thereof, while not using an application server to middle-man the data (e.g. MSs communicate with each other directly and wirelessly as peers). Application data fields shared between peer data processing systems (e.g. MSs) are preferably additionally available at a MS as AppTerm data (see below). A user has control for disabling or enabling which application data fields are shared. Privileges configured between MSs enforce desired effects for processing the data on MSs which send or receive the data.

A further advantage is to provide MSs with a wealth of location based enhanced applications without requiring a service. It is also an advantage to not require a service for geo-fence alerts, proactive content delivery, and nearby alerts, for example as described by server based U.S. Patent pending Ser. No. 11/207,080 ("System and Method for Anonymous Location Based Services", Johnson). Herein, alert processing, geo-fences and content is maintained at a MS for a) being processed at the MS when interacting directly with peer MSs; and b) being shared with peer MSs for being processed at peer MSs. Better performance of processing content delivery and providing alerts is achieved because it occurs at the MSs without any interoperability to some "middleman" service.

Another advantage is in leveraging the multi-threaded and wireless multi-wave, multi-frequency and multi-channel capability of the disclosed MS for RFID and RDS integration. RFID and RDS interfaces fit nicely in the LBX framework as described below.

A further advantage is for the MS to automatically, or upon user request, analyze a picture, or video stream frame, for the purpose of more confidently determining a MS location. User configurations are used to drive desired processing.

US 10,477,994 B2

17

Another advantage is for thoroughly maintaining and managing statistics and history information at a MS. Many options are supported for how, where, and when to save such information.

A further advantage is to provide Sudden Proximal User Interfaces (SPUIs) at a MS when detecting other data processing systems in the vicinity (e.g. another MS, a RFID device, a data processing system emulating a MS, or any other data processing system). A SPUI is a GUI for notifying a MS user that a remote data processing system of interest is in the vicinity, based on configured “in the vicinity” conditions. Presenting the SPUI at the MS can be triggered by charter configurations, application term (AppTerm) trigger configurations, or RFID trigger configurations. There are many applications for SPUI processing for saving MS users time from MS user interface interactions for common tasks, for example appliance and device interfaces. Authentication can be automated. Also, SPUIs save data from previous executions for defaulting data in a subsequent execution thereby preventing the burdening of a MS user from re-entering data to the MS that was already entered once previously. There are many applications that fit within the SPUI framework, some of which are described below.

Another advantage is for providing a user with the ability to manually request to send/transmit outbound data with options for customizing, such as: a WDR, a derivative of a WDR, a subset of a WDR, a user configured set of data, or any customized set of data. If a WDR or derivative/subset thereof is to be sent, the WDR may first be searched for at the MS with user specified search criteria and/or transmitted outbound according to user specified transmission criteria.

It is an advantage to provide a task monitor/trace interface for examining MS task status for current and past system states. The task monitor interface permits convenient contextual charter creation as desired by the user based on task status findings.

It is an advantage for providing generic application record sorting based on: MS whereabouts, whereabouts of a particular MS, whereabouts of others in the vicinity, or other WDR search criteria for sorting WDRs maintained at the MS where the sort is requested.

Another advantage is for providing one or more vicinity monitors for indicating MSs of interest that are nearby. The multi-threaded MS supports a plurality of vicinity monitors. A MS user configures criteria/conditions (i.e. expression) for a vicinity monitor for being compared to WDR information as it is received at the MS. The expression result (True/False) determines whether or not the MS that originated the WDR is to be monitored within the particular vicinity monitor. A polling or asynchronous event (e.g. as WDRs received) design may be used.

Another advantage is for automatic inventory management processing for inventory items that are in the vicinity of a MS at some point in time. A MS user can move to the whereabouts of particular items he desires to keep an inventory of for automatically managing the inventory by counting the current stock, performing orders for stocking, and tracking an order. The MS user can configure payment information for automatic order processing. Inventory items are enabled for inventory management in having an associated data processing system (e.g. (RFID tag, affixed/integrated MS, etc). A MS user can manually perform an order using the automatically determined inventory count information, or the order can be scheduled for automatic ordering (e.g. using a calendar entry). Inventory items can be ordered individually or as a group, perhaps as part of a group hierarchy. Typical uses are for managing the life of a typical

18

MS user: products stocked in kitchen pantry, refrigerator, freezer, closet, office, bathrooms, laundry room, office supply closet, or other areas of a MS user’s home, office or place of work.

Another advantage is for providing a MS user with a convenient resource mapping of privileges and charters between identities. For example, it could be tedious figuring out all the privileges, grants and charters which are granted to one MS user, and then granting those same rights to another MS user. Such a task is error prone and time consuming. Resource mapper functionality is provided wherein all rights (e.g. privileges) of one identifier can be assigned to another identifier in a single operation. The same rights can subsequently be removed as a single operation. A MS user has the ability to model granting privileges and charters to an identity (e.g. group), and then assign all of those, or remove all of those, in a single operation to other identifiers.

A further advantage is for different applications to be correlated through cross application addressing so that features or contexts of one application can be used to automatically affect features or contexts of another application. Identifiers used in context of one application are correlated to another application form. For example, an email application recipient address is correlated to the phone application caller id for the same MS in order to instantly (upon user request) show all emails associated to a person on an active phone call. The correlation occurs transparently without needing to know addressing details. There can be many identifier forms for correlation for a single MS depending on an application in use.

Further features and advantages of the disclosure, as well as the structure and operation of various embodiments of the disclosure, are described in detail below with reference to the accompanying drawings. In the drawings, like reference numbers generally indicate identical, functionally similar, and/or structurally similar elements. The drawing in which an element first appears is indicated by the leftmost digit(s) in the corresponding reference number, except that reference numbers 1 through 99 may be found on the first 4 drawings of FIGS. 1A through 1D, and FIG. 1F. Dashed outlines (e.g. process blocks, data record fields) may be used in the drawings to highlight, or indicate optional embodiments, for example depending on MS performance considerations. None of the drawings, discussions, or materials herein is to be interpreted as limiting to a particular embodiment. The broadest interpretation is intended. Other embodiments accomplishing same functionality are within the spirit and scope of this disclosure. It should be understood that information is presented by example and many embodiments exist without departing from the spirit and scope of this disclosure.

BRIEF DESCRIPTION OF THE DRAWINGS

There is no guarantee that there are descriptions in this specification for explaining every novel feature found in the drawings. The present disclosure will be described with reference to the accompanying drawings, wherein:

FIG. 1A depicts a preferred embodiment high level example componentization of a MS in accordance with the present disclosure;

FIG. 1B depicts a Location Based eXchanges (LBX) architectural illustration for discussing the present disclosure;

US 10,477,994 B2

19

FIG. 1C depicts a Location Based Services (LBS) architectural illustration for discussing prior art of the present disclosure;

FIG. 1D depicts a block diagram of a data processing system useful for implementing a MS, ILM, DLM, centralized server, or any other data processing system disclosed herein;

FIG. 1E depicts a network illustration for discussing various deployments of whereabouts processing aspects of the present disclosure;

FIG. 1F depicts a network illustration for discussing LBX character provided to a MS through user LBX configurations made;

FIG. 2A depicts an illustration for describing automatic location of a MS through the MS coming into range of a stationary cellular tower;

FIG. 2B depicts an illustration for describing automatic location of a MS through the MS coming into range of some stationary antenna;

FIG. 2C depicts an illustration for discussing an example of automatically locating a MS through the MS coming into range of some stationary antenna;

FIG. 2D depicts a flowchart for describing a preferred embodiment of a service whereabouts update event of an antenna in-range detected MS when MS location awareness is monitored by a stationary antenna or cell tower;

FIG. 2E depicts a flowchart for describing a preferred embodiment of an MS whereabouts update event of an antenna in-range detected MS when MS location awareness is monitored by the MS;

FIG. 2F depicts a flowchart for describing a preferred embodiment of a procedure for inserting a Whereabouts Data Record (WDR) to an MS whereabouts data queue;

FIG. 3A depicts a locating by triangulation illustration for discussing automatic location of a MS;

FIG. 3B depicts a flowchart for describing a preferred embodiment of the whereabouts update event of a triangulated MS when MS location awareness is monitored by some remote service;

FIG. 3C depicts a flowchart for describing a preferred embodiment of the whereabouts update event of a triangulated MS when MS location awareness is monitored by the MS;

FIG. 4A depicts a locating by GPS triangulation illustration for discussing automatic location of a MS;

FIG. 4B depicts a flowchart for describing a preferred embodiment of the whereabouts update event of a GPS triangulated MS;

FIG. 5A depicts a locating by stationary antenna triangulation illustration for discussing automatic location of a MS;

FIG. 5B depicts a flowchart for describing a preferred embodiment of the whereabouts update event of a stationary antenna triangulated MS;

FIG. 6A depicts a flowchart for describing a preferred embodiment of a service whereabouts update event of a physically or logically connected MS;

FIG. 6B depicts a flowchart for describing a preferred embodiment of a MS whereabouts update event of a physically or logically connected MS;

FIGS. 7A, 7B and 7C depict a locating by image sensory illustration for discussing automatic location of a MS;

FIG. 7D depicts a flowchart for describing a preferred embodiment of graphically locating a MS, for example as illustrated by FIGS. 7A through 7C;

FIG. 8A heterogeneously depicts a locating by arbitrary wave spectrum illustration for discussing automatic location of a MS;

20

FIG. 8B depicts a flowchart for describing a preferred embodiment of locating a MS through physically contacting the MS;

FIG. 8C depicts a flowchart for describing a preferred embodiment of locating a MS through a manually entered whereabouts of the MS;

FIG. 9A depicts a table for illustrating heterogeneously locating a MS;

FIG. 9B depicts a flowchart for describing a preferred embodiment of heterogeneously locating a MS;

FIGS. 10A and 10B depict an illustration of a Locatable Network expanse (LN-Expanse) for describing locating of an ILM with all DLMs;

FIG. 10C depicts an illustration of a Locatable Network expanse (LN-Expanse) for describing locating of an ILM with an ILM and DLM;

FIGS. 10D, 10E, and 10F depict an illustration of a Locatable Network expanse (LN-Expanse) for describing locating of an ILM with all ILMs;

FIGS. 10G and 10H depict an illustration for describing the infinite reach of a Locatable Network expanse (LN-Expanse) according to MSs;

FIG. 10I depicts an illustration of a Locatable Network expanse (LN-Expanse) for describing a supervisory service;

FIG. 11A depicts a preferred embodiment of a Whereabouts Data Record (WDR) 1100 for discussing operations of the present disclosure;

FIGS. 11B, 11C and 11D depict an illustration for describing various embodiments for determining the whereabouts of an MS;

FIG. 11E depicts an illustration for describing various embodiments for automatically determining the whereabouts of an MS;

FIG. 12 depicts a flowchart for describing an embodiment of MS initialization processing;

FIGS. 13A through 13C depict an illustration of data processing system wireless data transmissions over some wave spectrum;

FIG. 14A depicts a flowchart for describing a preferred embodiment of MS LBX configuration processing;

FIG. 14B depicts a continued portion flowchart of FIG. 14A for describing a preferred embodiment of MS LBX configuration processing;

FIG. 15A depicts a flowchart for describing a preferred embodiment of DLM role configuration processing;

FIG. 15B depicts a flowchart for describing a preferred embodiment of ILM role configuration processing;

FIG. 15C depicts a flowchart for describing a preferred embodiment of a procedure for Manage List processing;

FIG. 16 depicts a flowchart for describing a preferred embodiment of NTP use configuration processing;

FIG. 17 depicts a flowchart for describing a preferred embodiment of WDR maintenance processing;

FIG. 18 depicts a flowchart for describing a preferred embodiment of a procedure for variable configuration processing;

FIG. 19 depicts an illustration for describing a preferred embodiment multithreaded architecture of peer interaction processing of a MS in accordance with the present disclosure;

FIG. 20 depicts a flowchart for describing a preferred embodiment of MS whereabouts broadcast processing;

FIG. 21 depicts a flowchart for describing a preferred embodiment of MS whereabouts collection processing;

FIG. 22 depicts a flowchart for describing a preferred embodiment of MS whereabouts supervisor processing;

US 10,477,994 B2

21

FIG. 23 depicts a flowchart for describing a preferred embodiment of MS timing determination processing;

FIG. 24A depicts an illustration for describing a preferred embodiment of a thread request queue record;

FIG. 24B depicts an illustration for describing a preferred embodiment of a correlation response queue record;

FIG. 24C depicts an illustration for describing a preferred embodiment of a WDR request record;

FIG. 25 depicts a flowchart for describing a preferred embodiment of MS WDR request processing;

FIG. 26A depicts a flowchart for describing a preferred embodiment of MS whereabouts determination processing;

FIG. 26B depicts a flowchart for describing a preferred embodiment of processing for determining a highest possible confidence whereabouts;

FIG. 27A depicts a flowchart for describing a preferred embodiment of queue prune processing;

FIG. 27B depicts a flowchart for describing a preferred embodiment of setting confidence default values based on user experience;

FIG. 28 depicts a flowchart for describing a preferred embodiment of MS termination processing;

FIG. 29A depicts a flowchart for describing a preferred embodiment of a process for starting a specified number of threads in a specified thread pool;

FIG. 29B depicts a flowchart for describing a preferred embodiment of a procedure for terminating the process started by FIG. 29A;

FIGS. 30A through 30B depict a preferred embodiment BNF grammar for variables, variable instantiations and common grammar for BNF grammars of permissions, groups and charters;

FIG. 30C depicts a preferred embodiment BNF grammar for permissions and groups;

FIGS. 30D through 30E depict a preferred embodiment BNF grammar for charters;

FIGS. 31A through 31E depict a preferred embodiment set of command and operand candidates for Action Data Records (ADRs) facilitating discussing associated parameters of the ADRs of the present disclosure;

FIG. 32A depicts a preferred embodiment of a National Language Support (NLS) directive command cross reference;

FIG. 32B depicts a preferred embodiment of a NLS directive operand cross reference;

FIG. 33A depicts a preferred embodiment American National Standards Institute (ANSI) X.409 encoding of the BNF grammar of FIGS. 30A through 30B for variables, variable instantiations and common grammar for BNF grammars of permissions and charters;

FIG. 33B depicts a preferred embodiment ANSI X.409 encoding of the BNF grammar of FIG. 30C for permissions and groups;

FIGS. 33C-1 and 33C-2 (both hereinafter referred to as FIG. 33C) depict a preferred embodiment ANSI X.409 encoding of the BNF grammar of FIGS. 30D through 30E for charters;

FIGS. 34A through 34G depict preferred embodiment C programming source code header file contents, derived from the grammar of FIGS. 30A through 30E;

FIG. 35A depicts a preferred embodiment of a Granting Data Record (GDR) for discussing operations of the present disclosure, derived from the grammar of FIGS. 30A through 30E;

22

FIG. 35B depicts a preferred embodiment of a Grant Data Record (GRDTR) for discussing operations of the present disclosure, derived from the grammar of FIGS. 30A through 30E;

FIG. 35C depicts a preferred embodiment of a Generic Assignment Data Record (GADR) for discussing operations of the present disclosure, derived from the grammar of FIGS. 30A through 30E;

FIG. 35D depicts a preferred embodiment of a Privilege Data Record (PDR) for discussing operations of the present disclosure, derived from the grammar of FIGS. 30A through 30E;

FIG. 35E depicts a preferred embodiment of a Group Data Record (GRPDR) for discussing operations of the present disclosure, derived from the grammar of FIGS. 30A through 30E;

FIG. 36A depicts a preferred embodiment of a Description Data Record (DDR) for discussing operations of the present disclosure, derived from the grammar of FIGS. 30A through 30E;

FIG. 36B depicts a preferred embodiment of a History Data Record (HDR) for discussing operations of the present disclosure, derived from the grammar of FIGS. 30A through 30E;

FIG. 36C depicts a preferred embodiment of a Time specification Data Record (TDR) for discussing operations of the present disclosure, derived from the grammar of FIGS. 30A through 30E;

FIG. 36D depicts a preferred embodiment of a Variable Data Record (VDR) for discussing operations of the present disclosure, derived from the grammar of FIGS. 30A through 30E;

FIG. 37A depicts a preferred embodiment of a Charter Data Record (CDR) for discussing operations of the present disclosure, derived from the grammar of FIGS. 30A through 30E;

FIG. 37B depicts a preferred embodiment of an Action Data Record (ADR) for discussing operations of the present disclosure, derived from the grammar of FIGS. 30A through 30E;

FIG. 37C depicts a preferred embodiment of a Parameter Data Record (PARMDR) for discussing operations of the present disclosure, derived from the grammar of FIGS. 30A through 30E;

FIG. 37D depicts a preferred embodiment of Charters Starters schema for discussing operations of the present disclosure;

FIG. 38 depicts a flowchart for describing a preferred embodiment of MS permissions configuration processing;

FIGS. 39A through 39B depict flowcharts for describing a preferred embodiment of MS user interface processing for permissions configuration;

FIGS. 40A through 40B depict flowcharts for describing a preferred embodiment of MS user interface processing for grants configuration;

FIGS. 41A through 41B depict flowcharts for describing a preferred embodiment of MS user interface processing for groups configuration;

FIG. 42 depicts a flowchart for describing a preferred embodiment of a procedure for viewing MS configuration information of others;

FIG. 43 depicts a flowchart for describing a preferred embodiment of a procedure for configuring MS acceptance of data from other MSs;

FIG. 44A depicts a flowchart for describing a preferred embodiment of a procedure for sending MS data to another MS;

US 10,477,994 B2

23

FIG. 44B depicts a flowchart for describing a preferred embodiment of receiving MS configuration data from another MS;

FIG. 45A depicts a flowchart for describing a preferred embodiment of MS charters configuration processing;

FIG. 45B depicts a flowchart for describing a preferred embodiment of MS charter enablement and disablement processing;

FIGS. 46A through 46B depict flowcharts for describing a preferred embodiment of MS user interface processing for charters configuration;

FIGS. 47A through 47B depict flowcharts for describing a preferred embodiment of MS user interface processing for actions configuration;

FIGS. 48A through 48B depict flowcharts for describing a preferred embodiment of MS user interface processing for parameter information configuration;

FIG. 49A depicts an illustration for preferred permission data characteristics in the present disclosure LBX architecture;

FIG. 49B depicts an illustration for preferred charter data characteristics in the present disclosure LBX architecture;

FIGS. 50A through 50C depict an illustration of data processing system wireless data transmissions over some wave spectrum;

FIG. 51A depicts an example of a source code syntactical encoding embodiment of permissions, derived from the grammar of FIGS. 30A through 30E;

FIG. 51B depicts an example of a source code syntactical encoding embodiment of charters, derived from the grammar of FIGS. 30A through 30E;

FIG. 52 depicts another preferred embodiment C programming source code header file contents, derived from the grammar of FIGS. 30A through 30E;

FIG. 53 depicts a preferred embodiment of a Prefix Registry Record (PRR) for discussing operations of the present disclosure;

FIG. 54 depicts an example of an XML syntactical encoding embodiment of permissions and charters, derived from the BNF grammar of FIGS. 30A through 30E;

FIG. 55A depicts a flowchart for describing a preferred embodiment of MS user interface processing for Prefix Registry Record (PRR) configuration;

FIG. 55B depicts a flowchart for describing a preferred embodiment of Application Term (AppTerm) data modification;

FIG. 56 depicts a flowchart for appropriately processing an encoding embodiment of the BNF grammar of FIGS. 30A through 30E, in context for a variety of parser processing embodiments;

FIG. 57 depicts a flowchart for describing a preferred embodiment of WDR In-process Triggering Smarts (WITS) processing;

FIG. 58 depicts an illustration for granted data characteristics in the present disclosure LBX architecture;

FIG. 59 depicts a flowchart for describing a preferred embodiment of a procedure for enabling LBX features and functionality in accordance with a certain type of permissions;

FIG. 60 depicts a flowchart for describing a preferred embodiment of a procedure for performing LBX actions in accordance with a certain type of permissions;

FIG. 61 depicts a flowchart for describing a preferred embodiment of performing processing in accordance with configured charters;

24

FIG. 62 depicts a flowchart for describing a preferred embodiment of a procedure for performing an action corresponding to a configured command;

FIG. 63A depicts a flowchart for describing a preferred embodiment of a procedure for Send command action processing;

FIGS. 63B-1 through 63B-7 depicts a matrix describing how to process some varieties of the Send command;

FIG. 63C depicts a flowchart for describing one embodiment of a procedure for Send command action processing, as derived from the processing of FIG. 63A;

FIG. 64A depicts a flowchart for describing a preferred embodiment of a procedure for Notify command action processing;

FIGS. 64B-1 through 64B-4 depicts a matrix describing how to process some varieties of the Notify command;

FIG. 64C depicts a flowchart for describing one embodiment of a procedure for Notify command action processing, as derived from the processing of FIG. 64A;

FIG. 65A depicts a flowchart for describing a preferred embodiment of a procedure for Compose command action processing;

FIGS. 65B-1 through 65B-7 depicts a matrix describing how to process some varieties of the Compose command;

FIG. 65C depicts a flowchart for describing one embodiment of a procedure for Compose command action processing, as derived from the processing of FIG. 65A;

FIG. 66A depicts a flowchart for describing a preferred embodiment of a procedure for Connect command action processing;

FIGS. 66B-1 through 66B-2 depicts a matrix describing how to process some varieties of the Connect command;

FIG. 66C depicts a flowchart for describing one embodiment of a procedure for Connect command action processing, as derived from the processing of FIG. 66A;

FIG. 67A depicts a flowchart for describing a preferred embodiment of a procedure for Find command action processing;

FIGS. 67B-1 through 67B-13 depicts a matrix describing how to process some varieties of the Find command;

FIG. 67C depicts a flowchart for describing one embodiment of a procedure for Find command action processing, as derived from the processing of FIG. 67A;

FIG. 68A depicts a flowchart for describing a preferred embodiment of a procedure for Invoke command action processing;

FIGS. 68B-1 through 68B-5 depicts a matrix describing how to process some varieties of the Invoke command;

FIG. 68C depicts a flowchart for describing one embodiment of a procedure for Invoke command action processing, as derived from the processing of FIG. 68A;

FIG. 69A depicts a flowchart for describing a preferred embodiment of a procedure for Copy command action processing;

FIGS. 69B-1 through 69B-14 depicts a matrix describing how to process some varieties of the Copy command;

FIG. 69C depicts a flowchart for describing one embodiment of a procedure for Copy command action processing, as derived from the processing of FIG. 69A;

FIG. 70A depicts a flowchart for describing a preferred embodiment of a procedure for Discard command action processing;

FIGS. 70B-1 through 70B-11 depicts a matrix describing how to process some varieties of the Discard command;

FIG. 70C depicts a flowchart for describing one embodiment of a procedure for Discard command action processing, as derived from the processing of FIG. 70A;

US 10,477,994 B2

25

FIG. 71A depicts a flowchart for describing a preferred embodiment of a procedure for Move command action processing;

FIGS. 71B-1 through 71B-14 depicts a matrix describing how to process some varieties of the Move command;

FIG. 71C depicts a flowchart for describing one embodiment of a procedure for Move command action processing, as derived from the processing of FIG. 71A;

FIG. 72A depicts a flowchart for describing a preferred embodiment of a procedure for Store command action processing;

FIGS. 72B-1 through 72B-5 depicts a matrix describing how to process some varieties of the Store command;

FIG. 72C depicts a flowchart for describing one embodiment of a procedure for Store command action processing, as derived from the processing of FIG. 72A;

FIG. 73A depicts a flowchart for describing a preferred embodiment of a procedure for Administration command action processing;

FIGS. 73B-1 through 73B-7 depicts a matrix describing how to process some varieties of the Administration command;

FIG. 73C depicts a flowchart for describing one embodiment of a procedure for Administration command action processing, as derived from the processing of FIG. 73A;

FIG. 74A depicts a flowchart for describing a preferred embodiment of a procedure for Change command action processing;

FIG. 74C depicts a flowchart for describing one embodiment of a procedure for Change command action processing, as derived from the processing of FIG. 74A;

FIG. 75A depicts a flowchart for describing a preferred embodiment of a procedure for sending data to a remote MS;

FIG. 75B depicts a flowchart for describing a preferred embodiment of processing for receiving execution data from another MS;

FIG. 76A depicts a flowchart for describing a preferred embodiment of processing a special term information paste action at a MS;

FIG. 76B-1 illustrates a preferred embodiment of Application term interface processing;

FIG. 76B-2 illustrates an embodiment of Application term interface processing for applications not using a standardized LBX coding practice;

FIG. 76B-3 illustrates a preferred embodiment of charter invocation interface processing;

FIG. 76C illustrates a preferred embodiment of Application term shared memory records;

FIG. 76D depicts a flowchart for describing a preferred embodiment of processing for contextual charter creation;

FIG. 77 depicts a flowchart for describing a preferred embodiment of configuring data to be maintained to WDR Application Fields;

FIG. 78 depicts a simplified example of an XML syntactical encoding embodiment of a profile for the profile section of WDR Application Fields;

FIG. 79A illustrates a branch subset of a tree structure;

FIG. 79B illustrates a binary tree equivalent to the tree structure of FIG. 79A which is used to support XML tag tree traversal processing;

FIG. 79C depicts a preferred embodiment C programming source code structure for encoding a node in an internalized XML tree;

FIG. 79D depicts a flowchart for describing a preferred embodiment of a procedure for profile match operator evaluation;

26

FIG. 80A depicts a LBX application fields implementation status table;

FIGS. 80B-1 through 80B-4 (referred generally as FIG. 80B) depict some section descriptions of registered LBX application fields;

FIG. 80C depicts a flowchart for describing a preferred embodiment of a procedure for application fields section initialization processing;

FIG. 80D depicts a flowchart for describing a preferred embodiment of MS Radio Frequency Identification (RFID) probe processing;

FIG. 80E depicts a flowchart for describing a preferred embodiment of processing for receiving data from an RFID device;

FIG. 81A depicts a flowchart for describing a preferred embodiment of processing for configuring criteria used by a MS to graphically locate itself;

FIG. 81B depicts a flowchart for describing a preferred embodiment of processing for a MS to graphically locate itself;

FIG. 82A depicts a flowchart for describing a preferred embodiment of processing for maintaining LBX history;

FIG. 82B depicts a flowchart for describing a procedure to maintain information to LBX history;

FIG. 83A depicts a flowchart for describing a preferred embodiment of processing for configuring LBX statistics;

FIG. 83B depicts a flowchart for describing a procedure to maintain information to LBX statistics;

FIG. 84A depicts a flowchart for describing a preferred embodiment of processing for configuring service propagation;

FIG. 84B depicts a flowchart for describing a procedure to process application fields according to how they are enabled or disabled;

FIG. 85A depicts a preferred embodiment of a Service Directory Record (SDR) for discussing operations of the present disclosure;

FIG. 85B depicts a flowchart for describing a preferred embodiment of a procedure for processing a request for a propagated service;

FIG. 85C depicts a flowchart for describing an example embodiment of MS application processing relevant for interfacing to a propagated service;

FIG. 85D depicts a flowchart for describing a preferred embodiment of processing at a MS when receiving a request for a propagated service from a remote MS;

FIG. 85E depicts a flowchart for describing a preferred embodiment of processing for an executable that updates service directory information;

FIG. 86A depicts a flowchart for describing a preferred embodiment of processing for configuring the service informant;

FIG. 86B depicts a flowchart for describing a preferred embodiment procedure to provide service informant processing;

FIG. 86C depicts a preferred embodiment of a Service Informant Record (SIR) for discussing operations of the present disclosure;

FIG. 87A depicts a flowchart for describing a preferred embodiment of Sudden Proximal User Interface (SPUI) processing;

FIG. 87B illustrates different embodiments for discussing various application data processing systems which can be automatically controlled by a MS according to the present disclosure;

FIG. 87C depicts a flowchart for describing a remote data processing system application environment covering an infinite number of MS controllable applications;

FIG. 88A depicts a flowchart for describing a preferred embodiment of manually transmitting WDR information;

FIG. 88B depicts a flowchart for describing a preferred embodiment of MS task monitor processing;

FIG. 89A depicts a flowchart for describing a preferred embodiment of updating a MS global variable for the last time a MS input peripheral was acted upon by a MS user;

FIG. 90A depicts a flowchart for a preferred embodiment for processing the request to specify a map term;

FIG. 90B depicts a preferred embodiment of a Map Term Data Record (MTDR) for discussing operations of the present disclosure, derived from the grammar of FIGS. 30A through 30E;

FIGS. 91A through 91B depict preferred data schema embodiments of automated inventory management for discussing operations of the present disclosure;

FIG. 91C depicts a flowchart for a preferred embodiment for inventory management processing;

FIG. 91D depicts a flowchart for a preferred embodiment of automatically processing whereabouts of inventory items in the vicinity of a MS;

FIG. 92A depicts a flowchart for a preferred embodiment for inventory group management processing;

FIG. 92B depicts a flowchart for a preferred embodiment for automatic order processing of inventory items according to a schedule;

FIG. 93A depicts a flowchart for a preferred embodiment for payment method management processing;

FIG. 93B depicts a flowchart for a preferred embodiment for pending inventory order management processing;

FIG. 94A depicts a flowchart for a preferred embodiment of a procedure for automatically ordering inventory;

FIG. 94B depicts a flowchart for a preferred embodiment for order services management processing;

FIG. 95A depicts a preferred embodiment of a resource mapper record for resource mapper processing of the present disclosure;

FIG. 95B depicts a flowchart for a preferred embodiment for automatic resource mapper processing;

FIG. 96A depicts a flowchart for a preferred embodiment for automatic application sort index processing;

FIG. 96B illustrates an example application use of sort index processing;

FIG. 97A depicts a flowchart for a preferred embodiment for vicinity monitor configuration processing;

FIG. 97B depicts a preferred embodiment of a Vicinity Monitor Data Record (VMDR) for discussing operations of vicinity monitor processing; and

FIG. 97C depicts a flowchart for a preferred embodiment for vicinity monitor processing.

DETAILED DESCRIPTION OF THE INVENTION

With reference now to detail of the drawings, the present disclosure is described. Obvious error handling is omitted from the flowcharts in order to focus on the key aspects of the present disclosure. Obvious error handling includes database I/O errors, field validation errors, errors as the result of database table/data constraints or unique keys, data access errors, communications interface errors or packet collision, hardware failures, checksum validations, bit error detections/corrections, and any other error handling as well known to those skilled in the relevant art in context of this

disclosure. A semicolon may be used in flowchart blocks to represent, and separate, multiple blocks of processing within a single physical block. This allows simpler flowcharts with less blocks in the drawings by placing multiple blocks of processing description in a single physical block of the flowchart. Flowchart processing is intended to be interpreted in the broadest sense by example, and not for limiting methods of accomplishing the same functionality. Preferably, field validation in the flowcharts checks for SQL injection attacks, communications protocol sniff and hack attacks, preventing of spoofing MS addresses, syntactical appropriateness, and semantics errors where appropriate. Disclosed user interface processing and/or screenshots are also preferred embodiment examples that can be implemented in other ways without departing from the spirit and scope of this disclosure. Alternative user interfaces (since this disclosure is not to be limiting) will use similar mechanisms, but may use different mechanisms without departing from the spirit and scope of this disclosure.

Locational terms such as whereabouts, location, position, area, destination, perimeter, radius, geofence, situational location, or any other related two or three dimensional locational term used herein to described position(s) and/or locations and/or whereabouts is to be interpreted in the broadest sense. Location field 1100c may include an area (e.g. on earth), a point (e.g. on earth), or a three dimensional bounds in space. In another example, a radius may define a sphere in space, rather than a circle in a plane. In some embodiments, a planet field forms part of the location (e.g. Earth, Mars, etc as part of field 1100c) for which other location information (e.g. latitude and longitude on Mars also part of field 1100c) is relative. In some embodiments, elevations (or altitudes) from known locatable point(s), distances from origin(s) in the universe, etc. can denote where exactly is a point of three dimensional space, or three dimensional sphere, area, or solid, is located. That same point can provide a mathematical reference to other points of the solid area/region in space. Descriptions for angles, pitches, rotations, etc from some reference point(s) may be further provided. Three dimensional areas/regions include a conical shape, cubical shape, spherical shape, pyramidal shape, irregular shapes, or any other shape either manipulated with a three dimensional graphic interface, or with mathematical model descriptions. Areas/regions in space can be occupied by a MS, passed through (e.g. by a traveler) by a MS, or referenced through configuration by a MS. In a three dimensional embodiment, nearby/nearness is determined in terms of three dimensional information, for example, a spherical radius around one MS intersecting a spherical radius around another MS. In a two dimensional embodiment, nearby/nearness is determined in terms of two dimensional information, for example, a circular radius around one MS intersecting a circular radius around another MS. Points can be specified as a point in a x-y-z plane, a point in polar coordinates, or the like, perhaps the center of a planet (e.g. Earth) or the Sun, some origin in the Universe, or any other origin for distinctly locating three dimensional location(s), positions, or whereabouts in space. Elevation (e.g. for earth, or some other planet, etc) may be useful to the three dimensional point of origin, and/or for the three dimensional region in space. A region in space may also be specified with connecting x-y-z coordinates together to bound the three dimensional region in space. There are many methods for representing a location (field 1100c) without departing from the spirit and scope of this disclosure. MSs, for example as carried by users, can travel by airplane

US 10,477,994 B2

29

through three dimensional areas/regions in space, or travel under the sea through three dimensional regions in space.

Various embodiments of communications between MSs, or an MS and service(s), will share channels (e.g. frequencies) to communicate, depending on when in effect. Sharing a channel will involve carrying recognizable and processable signature to distinguish transmissions for carrying data. Other embodiments of communications between MSs, or an MS and service(s), will use distinct channels to communicate, depending on when in effect. The number of channels that can be concurrently listened on and/or concurrently transmitted on by a data processing system will affect which embodiments are preferred. The number of usable channels will also affect which embodiments are preferred. This disclosure avoids unnecessary detail in different communication channel embodiments so as to not obfuscate novel material. Independent of various channel embodiments within the scope and spirit of the present disclosure, MSs communicate with other MSs in a peer to peer manner, in some aspects like automated walkie-talkies.

Novel features disclosed herein need not be provided as all or none. Certain features may be isolated in some MS embodiments, or may appear as any subset of features and functionality in other embodiments.

Location Based eXchanges (LBX) Architecture

FIG. 1A depicts a preferred embodiment high level example componentization of a MS in accordance with the present disclosure. A MS 2 includes processing behavior referred to as LBX Character 4 and Other Character 32. LBX character 4 provides processing behavior causing MS 2 to take on the character of a Location Based Exchange (LBX) MS according to the present disclosure. Other Character 32 provides processing behavior causing MS to take on character of prior art MSs in context of the type of MS. Other character 32 includes at least other processing code 34, other processing data 36, and other resources 38, all of which are well known to those skilled in the art for prior art MSs. Other character 32 provides a MS user with a limited set of configurability and functionality. In some embodiments, LBX character 4 components may, or may not, make use of other character 32 components 34, 36, and 38. Other character 32 components may, or may not, make use of LBX character 4 components 6 through 30.

LBX character 4 preferably includes at least Peer Interaction Processing (PIP) code 6, Peer Interaction Processing (PIP) data 8, self management processing code 18, self management processing data 20, WDR queue 22, send queue 24, receive queue 26, service informant code 28, and LBX history 30. Peer interaction processing (PIP) code 6 comprises executable code in software, firmware, or hardware form for carrying out LBX processing logic of the present disclosure when interacting with another MS. Peer interaction processing (PIP) data 8 comprises data maintained in any sort of memory of MS 2, for example hardware memory, flash memory, hard disk memory, a removable memory device, or any other memory means accessible to MS 2. PIP data 8 contains intelligence data for driving LBX processing logic of the present disclosure when interacting with other MSs. Self management processing code 18 comprises executable code in software, firmware, or hardware form for carrying out the local user interface LBX processing logic of the present disclosure. Self management processing data 20 contains intelligence data for driving processing logic of the present disclosure as disclosed for locally maintained LBX features. WDR queue 22 contains

30

Whereabouts Data Records (WDRs) 1100, and is a First-In-First-Out (FIFO) queue when considering housekeeping for pruning the queue to a reasonable trailing history of inserted entries (i.e. remove stale entries). WDR queue 22 is preferably designed with the ability of queue entry retrieval processing similar to Standard Query Language (SQL) querying, wherein one or more entries can be retrieved by querying with a conditional match on any data field(s) of WDR 1100 and returning lists of entries in order by an ascending or descending key on one or any ascending/descending ordered list of key fields.

All disclosed queues (e.g. 22, 24, 26, 1980, 1990 (See FIG. 19), or any other queue) are implemented with an appropriate thread-safe means of queue entry peeking (makes copy of sought queue entry without removing), discarding, retrieval, insertion, and queue entry field sorted search processing. Queues are understood to have an associated implicit semaphore to ensure appropriate synchronous access to queue data in a multi-threaded environment to prevent data corruption and misuse. Such queue interfaces are well known in popular operating systems. In MS operating system environments which do not have an implicit semaphore protected queue scheme, queue accesses in the present disclosure flowcharts are to be understood to have a previous request to a queue-assigned semaphore lock prior to queue access, and a following release of the semaphore lock after queue access. Operating systems without semaphore control may use methods to achieve similar thread-safe synchronization functionality. Queue functionality may be accomplished with lists, arrays, databases (e.g. SQL) and other methodologies without departing from the spirit and scope of queue descriptions herein.

Queue 22 alternate embodiments may maintain a plurality of WDR queues which segregate WDRs 1100 by field(s) values to facilitate timely processing. WDR queue 22 may be at least two (2) separate queues: one for maintaining the MS 2 whereabouts, and one for maintaining whereabouts of other MSs. WDR queue 22 may be a single instance WDR 1100 in some embodiments which always contains the most current MS 2 whereabouts for use by MS 2 applications (may use a sister queue 22 for maintaining WDRs from remote MSs). At least one entry is to be maintained to WDR queue 22 at all times for MS 2 whereabouts.

Send queue 24 (Transmit (Tx) queue) is used to send communications data, for example as intended for a peer MS within the vicinity (e.g. nearby as indicated by maximum range 1306) of the MS 2. Receive queue 26 (Receive (Rx) queue) is used to receive communications data, for example from peer MSs within the vicinity (e.g. nearby as indicated by maximum range 1306) of the MS 2. Queues 24 and 26 may also each comprise a plurality of queues for segregating data thereon to facilitate performance in interfacing to the queues, in particular when different queue entry types and/or sizes are placed on the queue. A queue interface for sending/receiving data to/from the MS is optimal in a multi-threaded implementation to isolate communications transport layers to processing behind the send/receive queue interfaces, but alternate embodiments may send/receive data directly from a processing thread disclosed herein. Queues 22, 24, and/or 26 may be embodied as a purely data form, or SQL database, maintained at MS 2 in persistent storage, memory, or any other storage means. In some embodiments, queues 24 and 26 are not necessary since other character 32 will already have accessible resources for carrying out some LBX character 4 processing.

Queue embodiments may contain fixed length records, varying length records, pointers to fixed length records, or

US 10,477,994 B2

31

pointers to varying length records. If pointers are used, it is assumed that pointers may be dynamically allocated for record storage on insertions and freed upon record use after discards or retrievals.

As well known to those skilled in the art, when a thread sends on a queue **24** in anticipation of a corresponding response, there is correlation data in the data sent which is sought in a response received by a thread at queue **26** so the sent data is correlated with the received data. In a preferred embodiment, correlation is built using a round-robin generated sequence number placed in data for sending along with a unique MS identifier (MS ID). If data is not already encrypted in communications, the correlation can be encrypted. While the unique MS identifier (MS ID) may help the MS identify which (e.g. wireless) data is destined for it, correlation helps identify which data at the MS caused the response. Upon receipt of data from a responder at queue **26**, correlation processing uses the returned correlation (e.g. field **1100m**) to correlate the sent and received data. In preferred embodiments, the sequence number is incremented each time prior to use to ensure a unique number, otherwise it may be difficult to know which data received is a response to which data was sent, in particular when many data packets are sent within seconds. When the sequence number reaches a maximum value (e.g. $2^{**}32-1$), then it is round-robbined to 0 and is incremented from there all over again. This assures proper correlation of data between the MS and responders over time. There are other correlation schemes (e.g. signatures, random number generation, checksum counting, bit patterns, date/time stamp derivatives) to accomplish correlation functionality. If send and receive queues of Other Character **32** are used, then correlation can be used in a similar manner to correlate a response with a request (i.e. a send with a receipt).

There may be good reason to conceal the MS ID when transmitting it wirelessly. In this embodiment, the MS ID is a dependable and recognizable derivative (e.g. a pseudo MS ID) that can be detected in communications traffic by the MS having the pseudo MS ID, while concealing the true MS ID. This would conceal the true MS ID from would-be hackers sniffing wireless protocol. The derivative can always be reliably the same for simplicity of being recognized by the MS while being difficult to associate to a particular MS. Further still, a more protected MS ID (from would-be hackers that take time to deduce how an MS ID is scrambled) can itself be a dynamically changing correlation anticipated in forthcoming communications traffic, thereby concealing the real MS ID (e.g. phone number or serial number), in particular when anticipating traffic in a response, yet still useful for directing responses back to the originating MS (with the pseudo MS ID (e.g. correlation)). A MS would know which correlation is anticipated in a response by saving it to local storage for use until it becomes used (i.e. correlated in a matching response), or becomes stale. In another embodiment, a correlation response queue (like CR queue **1990**) can be deployed to correlate responses with requests that contain different correlations for pseudo MS IDs. In all embodiments, the MS ID (or pseudo MS ID) of the present disclosure should enable targeting communications traffic to the MS.

Service informant code **28** comprises executable code in software, firmware, or hardware form for carrying out of informing a supervisory service. The present disclosure does not require a connected web service, but there are features for keeping a service informed with activities of MS LBX.

32

Service informant code **28** can communicate as requested any data **8**, **20**, **22**, **24**, **26**, **30**, **36**, **38**, or any other data processed at MS **2**.

LBX history **30** contains historical data useful in maintaining at MS **2**, and possibly useful for informing a supervisory service through service informant code **28**. LBX History **30** preferably has an associated thread of processing for keeping it pruned to the satisfaction of a user of MS **2** (e.g. prefers to keep last 15 days of specified history data, and 30 days of another specified history data, etc). With a suitable user interface to MS **2**, a user may browse, manage, alter, delete, or add to LBX History **30** as is relevant to processing described herein. Service informant code **28** may be used to cause sending of an outbound email, SMS message, outbound data packet, or any other outbound communication in accordance with LBX of the MS.

PIP data **8** preferably includes at least permissions **10**, charters **12**, statistics **14**, and a service directory **16**. Permissions **10** are configured to grant permissions to other MS users for interacting the way the user of MS **2** desires for them to interact. Therefore, permissions **10** contain permissions granted from the MS **2** user to other MS users. In another embodiment, permissions **10** additionally, or alternatively, contain permissions granted from other MS users to the MS **2** user. Permissions are maintained completely local to the MS **2**. Charters **12** provide LBX behavior conditional expressions for how MSs should interact with MS **2**. Charters **12** are configured by the MS **2** user for other MS users. In another embodiment, charters **12** additionally, or alternatively, are configured by other MS users for the MS **2** user. Some charters expressions depend on permissions **10**. Statistics **14** are maintained at MS **2** for reflecting peer (MS) to peer (MS) interactions of interest that occurred at MS **2**. In another embodiment, statistics **14** additionally, or alternatively, reflect peer (MS) to peer (MS) interactions that occurred at other MSs, preferably depending on permissions **10**. Service informant code **28** may, or may not, inform a service of statistics **14** maintained. Service directory **16** includes routing entries for how MS **2** will find a sought service, or how another MS can find a sought service through MS **2**.

In some embodiments, any code (e.g. **6**, **18**, **28**, **34**, **38**) can access, manage, use, alter, or discard any data (e.g. **8**, **20**, **22**, **24**, **26**, **30**, **36**, **38**) of any other component in MS **2**. Other embodiments may choose to keep processing of LBX character **4** and other character **32** disjoint from each other. Rectangular component boundaries are logical component representations and do not have to delineate who has access to what. MS (also MSs) references discussed herein in context for the new and useful features and functionality disclosed is understood to be an MS **2** (MSs **2**).

FIG. 1B depicts a Location Based eXchanges (LBX) architectural illustration for discussing the present disclosure. LBX MSs are peers to each other for locational features and functionality. An MS **2** communicates with other MSs without requiring a service for interaction. For example, FIG. 1B depicts a wireless network **40** of five (5) MSs. Each is able to directly communicate with others that are in the vicinity (e.g. nearby as indicated by maximum range **1306**). In a preferred embodiment, communications are limited reliability wireless broadcast datagrams having recognizable data packet identifiers. In another embodiment, wireless communications are reliable transport protocols carried out by the MSs, such as TCP/IP. In other embodiments, usual communications data associated with other character **32** include new data (e.g. Communications Key **1304**) in transmissions for being recognized by MSs within

US 10,477,994 B2

33

the vicinity. For example, as an MS conventionally communicates, LBX data is added to the protocol so that other MSs in the vicinity can detect, access, and use the data. The advantage to this is that as MSs use wireless communications to carry out conventional behavior, new LBX behavior is provided by simply incorporating additional information (e.g. Communications Key 1304) to existing communications.

Regardless of the embodiment, an MS 2 can communicate with any of its peers in the vicinity using methods described below. Regardless of the embodiment, a communication path 42 between any two MSs is understood to be potentially bidirectional, but certainly at least unidirectional. The bidirectional path 42 may use one communications method for one direction and a completely different communications method for the other, but ultimately each can communicate to each other. When considering that a path 42 comprises two unidirectional communications paths, there are $N*(N-1)$ unidirectional paths for N MSs in a network 40. For example, 10 MSs results in 90 (i.e. $10*9$) one way paths of communications between all 10 MSs for enabling them to talk to each other. Sharing of the same signaling channels is preferred to minimize the number of MS threads listening on distinct channels. Flowcharts are understood to process at incredibly high processing speeds, in particular for timely communications processing. While the MSs are communicating wirelessly to each other, path 42 embodiments may involve any number of intermediary systems or communications methods, for example as discussed below with FIG. 1E.

FIG. 1C depicts a Location Based Services (LBS) architectural illustration for discussing prior art of the present disclosure. In order for a MS to interact for LBS with another MS, there is service architecture 44 for accomplishing the interaction. For example, to detect that MS 1 is nearby MS N, the service is indispensably involved in maintaining data and carrying out processing. For example, to detect that MS 1 is arriving to, or departing from, a geofenced perimeter area configured by MS N, the service was indispensably involved in maintaining data and carrying out processing. For example, for MS N to locate MS 1 on a live map, the service was indispensably involved in maintaining data and carrying out processing. In another example, to grant and revoke permissions from MS 1 to MS N, the service was indispensably involved in maintaining data and carrying out processing. While it is advantageous to require a single bidirectional path 46 for each MS (i.e. two unidirectional communications paths; $(2*N)$ unidirectional paths for N MSs), there are severe requirements for service(s) when there are lots of MSs (i.e. when N is large). Wireless MSs have advanced beyond cell phones, and are capable of housing significant parallel processing, processing speed, increased wireless transmission speeds and distances, increased memory, and richer features.

FIG. 1D depicts a block diagram of a data processing system useful for implementing a MS, ILM, DLM, centralized server, or any other data processing system described herein. An MS 2 is a data processing system 50. Data processing system 50 includes at least one processor 52 (e.g. Central Processing Unit (CPU)) coupled to a bus 54. Bus 54 may include a switch, or may in fact be a switch 54 to provide dedicated connectivity between components of data processing system 50. Bus (and/or switch) 54 is a preferred embodiment coupling interface between data processing system 50 components. The data processing system 50 also includes main memory 56, for example, random access memory (RAM). Memory 56 may include multiple memory

34

cards, types, interfaces, and/or technologies. The data processing system 50 may include secondary storage devices 58 such as persistent storage 60, and/or removable storage device 62, for example as a compact disk, floppy diskette, USB flash, or the like, also connected to bus (or switch) 54. In some embodiments, persistent storage devices could be remote to the data processing system 50 and coupled through an appropriate communications interface. Persistent storage 60 may include flash memory, disk drive memory, magnetic, charged, or bubble storage, and/or multiple interfaces and/or technologies, perhaps in software interface form of variables, a database, shared memory, etc.

The data processing system 50 may also include a display device interface 64 for driving a connected display device (not shown). The data processing system 50 may further include one or more input peripheral interface(s) 66 to input devices such as a keyboard, keypad, Personal Digital Assistant (PDA) writing implements, touch interfaces, mouse, voice interface, or the like. User input ("user input", "user events" and "user actions" used interchangeably) to the data processing system are inputs accepted by the input peripheral interface(s) 66. The data processing system 50 may still further include one or more output peripheral interface(s) 68 to output devices such as a printer, facsimile device, or the like. Output peripherals may also be available via an appropriate interface.

Data processing system 50 will include communications interface(s) 70 for communicating to another data processing system 72 via analog signal waves, digital signal waves, infrared proximity, copper wire, optical fiber, or other wave spectrums described herein. A MS may have multiple communications interfaces 70 (e.g. cellular connectivity, 802.x, etc). Other data processing system 72 may be an MS. Other data processing system 72 may be a service. Other data processing system 72 is a service data processing system when MS 50 communicates to other data processing system 72 by way of service informant code 28. In any case, the MS and other data processing system are said to be interoperating when communicating.

Data processing system programs (also called control logic) may be completely inherent in the processor(s) 52 being a customized semiconductor, or may be stored in main memory 56 for execution by processor(s) 52 as the result of a read-only memory (ROM) load (not shown), or may be loaded from a secondary storage device into main memory 56 for execution by processor(s) 52. Such programs, when executed, enable the data processing system 50 to perform features of the present disclosure as discussed herein. Accordingly, such data processing system programs represent controllers of the data processing system.

In some embodiments, the disclosure is directed to a control logic program product comprising at least one processor 52 having control logic (software, firmware, hardware microcode) stored therein. The control logic, when executed by processor(s) 52, causes the processor(s) 52 to provide functions of the disclosure as described herein. In another embodiment, this disclosure is implemented primarily in hardware, for example, using a prefabricated component state machine (or multiple state machines) in a semiconductor element such as a processor 52.

Those skilled in the art will appreciate various modifications to the data processing system 50 without departing from the spirit and scope of this disclosure. A data processing system, and more particularly a MS, preferably has capability for many threads of simultaneous processing which provide control logic and/or processing. These threads can be embodied as time sliced threads of processing

US 10,477,994 B2

35

on a single hardware processor, multiple processors, multi-core processors, Digital Signal Processors (DSPs), or the like, or combinations thereof. Such multi-threaded processing can concurrently serve large numbers of concurrent MS tasks. Concurrent processing may be provided with distinct hardware processing and/or as appropriate software driven time-sliced thread processing. Those skilled in the art recognize that having multiple threads of execution on an MS is accomplished in many different ways without departing from the spirit and scope of this disclosure. This disclosure strives to deploy software to existing MS hardware configurations, but the disclosed software can be deployed as burned-in microcode to new hardware of MSs.

Data processing aspects of drawings/flowcharts are preferably multi-threaded so that many MSs and applicable data processing systems are interfaced with in a timely and optimal manner. Data processing system 50 may also include its own clock mechanism (not shown), if not an interface to an atomic clock or other clock mechanism, to ensure an appropriately accurate measurement of time in order to appropriately carry out processing described below. In some embodiments, Network Time Protocol (NTP) is used to keep a consistent universal time for MSs and other data processing systems in communications with MSs. This is most advantageous to prevent unnecessary round-tripping of data between data processing systems to determine timing (e.g. Time Difference of Arrival (TDOA)) measurements. A NTP synchronized date/time stamp maintained in communications is compared by a receiving data processing system for comparing with its own NTP date/time stamp to measure TOA (time of arrival (i.e. time taken to arrive)). Of course, in the absence of NTP used by the sender and receiver, TOA is also calculated in a bidirectional transmission using correlation. In this disclosure, TOA measurements from one location technology are used for triangulating with TOA measurements from another location technology, not just for determining "how close". Therefore, TDOA terminology is generally used herein to refer to the most basic TOA measurement of a wave spectrum signal being the difference between when it was sent and when it was received. TDOA is also used to describe using the difference of such measurements to locate (triangulate). NTP use among participating systems has the advantage of a single unidirectional broadcast data packet containing all a receiving system requires to measure TDOA, by knowing when the data was sent (date/time stamp in packet) and when the data was received (signal detected and processed by receiving system). A NTP clock source (e.g. atomic clock) used in a network is to be reasonably granular to carry out measurements, and ensures participating MSs are updated timely according to anticipated time drifts of their own clocks. MS clocks should maintain time as accurately as possible to minimize drift and minimize how often resynchronization with a NTP clock source is required. There are many well known methods for accomplishing NTP, some which require dedicated thread(s) for NTP processing, and some which use certain data transmitted to and from a source to keep time in synch.

Those skilled in the art recognize that NTP accuracy depends on participating MS clocks and processing timing, as well as time server source(s). Radio wave connected NTP time server(s) is typically accurate to as granular as 1 millisecond. Global Positioning System (GPS) time servers provide accuracy as granular as 50 microseconds. GPS timing receivers provide accuracy to around 100 nanoseconds, but this may be reduced by timing latencies in time server operating systems. With advancements in hardware,

36

microcode, and software, obvious improvements are being made to NTP. In NTP use embodiments of this disclosure, an appropriate synchronization of time is used for functional interoperability between MSs and other data processing systems using NTP. NTP is not required in this disclosure, but it is an advantage when in use.

LBX Directly Located Mobile Data Processing Systems (DLMs)

FIG. 1E depicts a network illustration for discussing various deployments of whereabouts processing aspects of the present disclosure. In some embodiments, a cellular network cluster 102 and cellular network cluster 104 are parts of a larger cellular network. Cellular network cluster 102 contains a controller 106 and a plurality of base stations, shown generally as base stations 108. Each base station covers a single cell of the cellular network cluster, and each base station 108 communicates through a wireless connection with the controller 106 for call processing, as is well known in the art. Wireless devices communicate via the nearest base station (i.e. the cell the device currently resides in), for example base station 108b. Roaming functionality is provided when a wireless device roams from one cell to another so that a session is properly maintained with proper signal strength. Controller 106 acts like a telephony switch when a wireless device roams across cells, and it communicates with controller 110 via a wireless connection so that a wireless device can also roam to other clusters over a larger geographical area. Controller 110 may be connected to a controller 112 in a cellular cluster through a physical connection, for example, copper wire, optical fiber, or the like. This enables cellular clusters to be great distances from each other. Controller 112 may in fact be connected with a physical connection to its base stations, shown generally as base stations 114. Base stations may communicate directly with the controller 112, for example, base station 114e. Base stations may communicate indirectly to the controller 112, for example base station 114a by way of base station 114d. It is well known in the art that many options exist for enabling interoperating communications between controllers and base stations for the purpose of managing a cellular network. A cellular network cluster 116 may be located in a different country. Base controller 118 may communicate with controller 110 through a Public Service Telephone Network (PSTN) by way of a telephony switch 120, PSTN 122, and telephony switch 124, respectively. Telephony switch 120 and telephony switch 124 may be private or public. In one cellular network embodiment of the present disclosure, the services execute at controllers, for example controller 110. In some embodiments, the MS includes processing that executes at a wireless device, for example mobile laptop computer 126, wireless telephone 128, a personal digital assistant (PDA) 130, an iPhone 170, or the like. As the MS moves about, positional attributes are monitored for determining location. The MS may be handheld, or installed in a moving vehicle. Locating a wireless device using wireless techniques such as Time Difference of Arrival (TDOA) and Angle Of Arrival (AOA) are well known in the art. The service may also execute on a server computer accessible to controllers, for example server computer 132, provided an appropriate timely connection exists between cellular network controller(s) and the server computer 132. Wireless devices (i.e. MSs) are preferably known by a unique identifier, for example a phone number, caller id, device identifier, or like appropriate unique handle.

US 10,477,994 B2

37

In another embodiment of the present disclosure, GPS satellites such as satellite **134**, satellite **136**, and satellite **138** provide information, as is well known in the art, to GPS devices on earth for triangulation locating of the GPS device. In this embodiment, a MS has integrated GPS functionality so that the MS monitors its positions. The MS is preferably known by a unique identifier, for example a phone number, caller id, device identifier, or like appropriate unique handle (e.g. network address).

In yet another embodiment of the present disclosure, a physically connected device, for example, telephone **140**, computer **142**, PDA **144**, telephone **146**, and fax machine **148**, may be newly physically connected to a network. Each is a MS, although the mobility is limited. Physical connections include copper wire, optical fiber, USB, or any other physical connection, by any communications protocol thereon. Devices are preferably known by a unique identifier, for example a phone number, caller id, device identifier, physical or logical network address, or like appropriate unique handle. The MS is detected for being newly located when physically connected. A service can be communicated to upon detecting connectivity. The service may execute at an Automatic Response Unit (ARU) **150**, a telephony switch, for example telephony switch **120**, a web server **152** (for example, connected through a gateway **154**), or a like data processing system that communicates with the MS in any of a variety of ways as well known to those skilled the art. MS detection may be a result of the MS initiating a communication with the service directly or indirectly. Thus, a user may connect his laptop to a hotel network, initiate a communication with the service, and the service determines that the user is in a different location than the previous communication. A local area network (LAN) **156** may contain a variety of connected devices, each an MS that later becomes connected to a local area network **158** at a different location, such as a PDA **160**, a server computer **162**, a printer **164**, an internet protocol telephone **166**, a computer **168**, or the like. Hard copy presentation could be made to printer **164** and fax **148**.

Current technology enables devices to communicate with each other, and other systems, through a variety of heterogeneous system and communication methods. Current technology allows executable processing to run on diverse devices and systems. Current technology allows communications between the devices and/or systems over a plethora of methodologies at close or long distance. Many technologies also exist for automatic locating of devices. It is well known how to have an interoperating communications system that comprises a plurality of individual systems communicating with each other with one or more protocols. As is further known in the art of developing software, executable processing of the present disclosure may be developed to run on a particular target data processing system in a particular manner, or customized at install time to execute on a particular data processing system in a particular manner.

FIG. 1F depicts a network illustration for discussing LBX character **4** provided to a MS through LBX configurations made, for example with permissions **10** and/or charters **12**. FIG. 1F exemplifies FIG. 1B in how user configurations provide wits and a unique personality to a MS. LBX character **4** wits (see WITS below) enable a vast and diverse set of processing behavior for location based processing, even for identically manufactured MSs having identically available applications for use. Every MS **2** can be very different and distinguished from other MSs **2** depending on permissions **10** and charters **12** which are configured for driving WITS processing. For example, a MS **2p** with “Hog”

38

LBX character contains user configurations for selfishly leveraging the LN-expanse for being located while never providing information for others to be located. Hog MS **2p** contains configurations that are rich and deep in functionality for the user of MS **2p**, but provide little functionality for other MS users. A MS **2q** with “Monkey” LBX character contains configurations for “fun and games” which are suitable for interacting with other MSs for primarily entertainment and playful purposes. Monkey MS **2q** contains configurations that provide enjoyment to the MS user and his peers. A MS **2r** with “Dog” LBX character contains configurations for “being everyone’s best friend” whereby MS **2r** maintains configurations for helping others in accordance with any requests made on behalf of peer MSs. For example, the user of MS **2r** is willing to unquestionably create configurations to keep LBX peers happy and to facilitate locational applications at other MSs. A MS **2s** with “Cow” LBX character contains configurations for “existing to contribute” to the LN-Expanse by maintaining configurations for facilitating the locating of other MSs, and to interact with other MSs for the purpose of supporting locational applications at other MSs without being solicited for support. A MS **2t** with “Tiger” LBX character contains user configurations which are “strictly business” and suitable for interacting with other MSs for primarily locational business purposes. Tiger MS **2** contains configurations for allowing business associates to interact, for example for letting a boss and team member know whereabouts, or alerting business associates of being nearby, or for automatically performing charter actions for the purpose of improving business activities. The richness of locational features and functionality provided by the LBX architecture enables a MS user to configure an infinite set of LBX character **4** for characterizing a MS and how it interacts with other MSs. Users exploit their own creativity for how their MSs should behave and what personalities their MS should have. The user’s MS becomes a broader reaching, and more impacting, personification of a user’s moving presence.

In some embodiments, an administrator or authorized user (e.g. parent) configures the MS for intended LBX character and use by the main MS user (e.g. child). Credentials such as a password, access code, user identifier and password, etc, or other authorization scheme may be used when accessing a disclosed configuration interface to limit configurability to certain users, types of users, or users with certain privileges.

FIG. 2A depicts an illustration for describing automatic location of a MS, for example a DLM **200**, through the MS coming into range of a stationary cellular tower. A DLM **200**, or any of a variety of MSs, travels within range of a cell tower, for example cell tower **108b**. The known cell tower location is used to automatically detect the location of the DLM **200**. In fact, any DLM that travels within the cell served by cell tower **108b** is identified as the location of cell tower **108b**. The confidence of a location of a DLM **200** is low when the cell coverage of cell tower **108b** is large. In contrast, the confidence of a location of a DLM **200** is higher when the cell coverage of cell tower **108b** is smaller. However, depending on the applications locating DLMs using this method, the locating can be quite acceptable. Location confidence is improved with a TDOA measurement for the elapsed time of communication between DLM **200** and cell tower to determine how close the MS is to the cell tower. Cell tower **108b** can process all locating by itself, or with interoperability to other services as connected to cell tower **108b** in FIG. 1E. Cell tower **108b** can communicate the location of DLM **200** to a service, to the DLM **200**, to

US 10,477,994 B2

39

other MSs within its coverage area, any combination thereof, or to any connected data processing system, or MS, of FIG. 1E.

FIG. 2B depicts an illustration for describing automatic location of a MS, for example a DLM 200, through the MS coming into range of some stationary antenna. DLM 200, or any of a variety of MSs, travels within range of a stationary antenna 202 that may be mounted to a stationary object 204. The known antenna location is used to automatically detect the location of the DLM 200. In fact, any DLM that travels within the coverage area served by antenna 202 is identified as the location of antenna 202. The confidence of a location of a DLM 200 is low when the antenna coverage area of antenna 202 is large. In contrast, the confidence of a location of a DLM 200 is higher when the antenna coverage area of antenna 202 is smaller. However, depending on the applications locating DLMs using this method, the locating can be quite acceptable. Location confidence is improved with a TDOA measurement for the elapsed time of communication between DLM 200 and a particular antenna to determine how close the MS is to the antenna. Antenna 202 can process all locating by itself (with connected data processing system (not shown) as well known to those skilled in the art), or with interoperability to other services as connected to antenna 202, for example with connectivity described in FIG. 1E. Antenna 202 can be used to communicate the location of DLM 200 to a service, to the DLM 200, to other MSs within its coverage area, any combination thereof, or to any connected data processing system, or MS, of FIG. 1E.

FIG. 2C depicts an illustration for discussing an example of automatically locating a MS, for example a DLM 200, through the MS coming into range of some stationary antenna. DLM 200, or any of a variety of MSs, travels within range of a stationary antenna 212 that may be mounted to a stationary object, such as building 210. The known antenna location is used to automatically detect the location of the DLM 200. In fact, any DLM that travels within the coverage area served by antenna 212 is identified as the location of antenna 212. The confidence of a location of a DLM 200 is low when the antenna coverage area of antenna 212 is large. In contrast, the confidence of a location of a DLM 200 is higher when the antenna coverage area of antenna 212 is smaller. However, depending on the applications locating DLMs using this method, the locating can be quite acceptable. Location confidence is improved with a TDOA measurement as described above. Antenna 212 can process all locating by itself (with connected data processing system (not shown) as well known to those skilled in the art), or with interoperability to other services as connected to antenna 212, for example with connectivity described in FIG. 1E. Antenna 212 can be used to communicate the location of DLM 200 to a service, to the DLM 200, to other MSs within its coverage area, any combination thereof, or to any connected data processing system, or MS, of FIG. 1E.

Once DLM 200 is within the building 210, a strategically placed antenna 216 with a desired detection range within the building is used to detect the DLM 200 coming into its proximity. Wall breakout 214 is used to see the antenna 216 through the building 210. The known antenna 216 location is used to automatically detect the location of the DLM 200. In fact, any DLM that travels within the coverage area served by antenna 216 is identified as the location of antenna 216. The confidence of a location of a DLM 200 is low when the antenna coverage area of antenna 216 is large. In contrast, the confidence of a location of a DLM 200 is higher when the antenna coverage area of antenna 216 is smaller. Travels of DLM 200 can be limited by objects, pathways, or

40

other limiting circumstances of traffic, to provide a higher confidence of location of DLM 200 when located by antenna 216, or when located by any locating antenna described herein which detects MSs coming within range of its location. Location confidence is improved with a TDOA measurement as described above. Antenna 216 can process all locating by itself (with connected data processing system (not shown) as well known to those skilled in the art), or with interoperability to other services as connected to antenna 216, for example with connectivity described in FIG. 1E. Antenna 216 can be used to communicate the location of DLM 200 to a service, to the DLM 200, to other MSs within its coverage area, any combination thereof, or to any connected data processing system, or MS, of FIG. 1E. Other in-range detection antennas of a FIG. 2C embodiment may be strategically placed to facilitate warehouse operations such as in Kubler et al.

FIG. 2D depicts a flowchart for describing a preferred embodiment of a service whereabouts update event of an antenna in-range detected MS, for example a DLM 200, when MS location awareness is monitored by a stationary antenna, or cell tower (i.e. the service thereof). FIGS. 2A through 2C location detection processing are well known in the art. FIG. 2D describes relevant processing for informing MSs of their own whereabouts. Processing begins at block 230 when a MS signal deserving a response has been received and continues to block 232 where the antenna or cell tower service has authenticated the MS signal. A MS signal can be received for processing by blocks 230 through 242 as the result of a continuous, or pulsed, broadcast or beaconing by the MS (FIG. 13A), perhaps as part of usual communication protocol in progress for the MS (FIG. 13A usual data 1302 with embedded Communications Key (CK) 1304), or an MS response to continuous, or pulsed, broadcast or beaconing via the service connected antenna (FIG. 13C). MS and/or service transmission can be appropriately correlated for a response (as described above) which additionally facilitates embodiments using TDOA measurements (time of communications between the MS and antenna, or cell tower) to determine at least how close is the MS in range (or use in conjunction with other data to triangulate the MS location). The MS is preferably authenticated by a unique MS identifier such as a phone number, address, name, serial number, or any other unique handle to the MS. In this, and any other embodiments disclosed, an MS may be authenticated using a group identifier handle indicating membership to a supported/known group deserving further processing. Authentication will preferably consult a database for authenticating that the MS is known. Block 232 continues to block 234 where the signal received is immediately responded back to the MS, via the antenna, containing at least correlation along with whereabouts information for a Whereabouts Data Record (WDR) 1100 associated with the antenna (or cell tower). Thereafter, the MS receives the correlated response containing new data at block 236 and completes a local whereabouts data record 1100 (i.e. WDR 1100) using data received along with other data determined by the MS.

In another embodiment, blocks 232 through 234 are not required. A service connected antenna (or cell tower) periodically broadcasts its whereabouts (WDR info (e.g. FIG. 13C)) and MSs in the vicinity use that directly at block 236. The MS can choose to use only the confidence and location provided, or may determine a TDOA measurement for determining how close it is. If the date/time stamp field 1100b indicates NTP is in use by the service, and the MS is also using NTP, then a TDOA measurement can be deter-

US 10,477,994 B2

41

mined using the one unidirectional broadcast via the antenna by using the date/time stamp field **1100b** received with when the WDR information was received by the MS (subtract time difference and use known wave spectrum for distance). If either the service or MS is not NTP enabled, then a bi-

directional correlated data flow between the service and MS is used to assess a TDOA measurement in terms of time of the MS. One embodiment provides the TDOA measurement from the service to the MS. Another embodiment calculates the TDOA measurement at the MS.

Network Time protocol (NTP) can ensure MSs have the same atomic clock time as the data processing systems driving antennas (or cell towers) they will encounter. Then, date/time stamps can be used in a single direction (unidirectional) broadcast packet to determine how long it took to arrive to/from the MS. In an NTP embodiment, the MS (FIG. 13A) and/or the antenna (FIG. 13C) sends a date/time stamp in the pulse, beacon, or protocol. Upon receipt, the antenna (or cell tower) service data processing system communicates how long the packet took from an MS to the antenna (or cell tower) by comparing the date/time stamp in the packet and a date/time stamp of when it was received. The service may also set the confidence value, before sending WDR information to the MS. Similarly, an MS can compare a date/time stamp in the unidirectional broadcast packet sent from a locating service (FIG. 13C) with when received by the MS. So, NTP facilitates TDOA measurements in a single broadcast communication between systems through incorporation to usual communications data **1302** with a date/time stamp in Communications Key (CK) **1304**, or alternatively in new data **1302**. Similarly, NTP facilitates TDOA measurement in a single broadcast communication between systems through incorporation to usual communications data **1312** with a date/time stamp in Communications Key (CK) **1314**, or alternatively in new data **1312**.

The following template is used in this disclosure to highlight field settings. See FIG. 11A descriptions. Fields are set to the following upon exit from block **236**:

MS ID field **1100a** is preferably set with: Unique MS identifier of the MS invoking block **240**. This field is used to uniquely distinguish this MS WDRs on queue **22** from other originated WDRs.

DATE/TIME STAMP field **1100b** is preferably set with: Date/time stamp for WDR completion at block **236** to the finest granulation of time achievable by the MS. The NTP use indicator is set appropriately.

LOCATION field **1100c** is preferably set with: Location of stationary antenna (or cell tower) as communicated by the service to the MS.

CONFIDENCE field **1100d** is preferably set with: The same value (e.g. 76) for any range within the antenna (or cell tower), or may be adjusted using the TDOA measurement (e.g. amount of time detected by the MS for the response at block **234**). The longer time it takes between the MS sending a signal detected at block **232** and the response with data back received by the MS (block **234**), the less confidence there is for being located because the MS must be a larger distance from the antenna or cell tower. The less time it takes between the MS sending a signal detected at block **232** and the response with data back, the more confidence there is for being located because the MS must be a closer distance to the antenna or cell tower. Confidence values are standardized for all location technologies. In some embodiments of FIG. 2D processing, a confidence value can be set for 1 through 100 (1 being lowest confidence and 100 being highest confidence) wherein a unit of measurement between the MS and antenna (or cell tower) is used directly for the

42

confidence value. For example, 20 meters is used as the unit of measurement. For each unit of 20 meters distance determined by the TDOA measurement, assign a value of 1, up to a worst case of 100 (i.e. 2000 meters). Round the 20 meter unit of distance such that 0 meters to <25 meters is 20 meters (i.e. 1 unit of measurement), 26 meters to <45 meters to is 40 meters (i.e. 2 units of measurement), and so on. Once the number of units is determined, subtract that number from 101 for the confidence value (i.e. 1 unit=confidence value 101, 20 units=confidence value 81; 100 units or greater=confidence value of 1). Yet another embodiment will use a standard confidence value for this “coming in range” technology such as 76 and then further increase or decrease the confidence using the TDOA measurement.

Many embodiments exist for quantifying a higher versus lower confidence. In any case, a confidence value (e.g. 76) is determined by the MS, service, or both (e.g. MS uses TDOA measurement to modify confidence sent by service). LOCATION TECHNOLOGY field **1100e** is preferably set with: “Server Antenna Range” for an antenna detecting the MS, and is set to “Server Cell Range” for a cell tower detecting the MS. The originator indicator is set to DLM. LOCATION REFERENCE INFO field **1100f** is preferably set with: The period of time for communications between the antenna and the MS (a TDOA measurement), if known; a communications signal strength, if available; wave spectrum used (e.g. from MS receive processing), if available; particular communications interface **70**, if available. The TDOA measurement may be converted to a distance using wave spectrum information. The values populated here should have already been factored into the confidence value at block **236**.

COMMUNICATIONS REFERENCE INFO field **1100g** is preferably set with: Parameters uniquely identifying a/the service (e.g. antenna (or cell tower)) and how to best communicate with it again, if available. May not be set, regardless if received from the service.

SPEED field **1100h** is preferably set with: Data received by MS at block **234**, if available.

HEADING field **1100i** is preferably set with: Data received by MS at block **234**, if available.

ELEVATION field **1100j** is preferably set with: data received by MS at block **234**, if available. Elevation field **1100j** is preferably associated with the antenna (or cell tower) by the elevation/altitude of the antenna (or cell tower).

APPLICATION FIELDS field **1100k** is preferably set with: Data received at block **234** by the MS, or set by data available to the MS, or set by both the locating service for the antenna (or cell tower) and the MS itself. Application fields include, and are not limited to, MS navigation APIs in use, social web site identifying information, application information for applications used, accessed, or in use by the MS, or any other information complementing whereabouts of the MS.

CORRELATION FIELD **1100m** is preferably set with: Not Applicable (i.e. not maintained to queue **22**).

SENT DATE/TIME STAMP field **1100n** is preferably set with: Not Applicable (i.e. not maintained to queue **22**).

RECEIVED DATE/TIME STAMP field **1100p** is preferably set with: Not Applicable (i.e. not maintained to queue **22**).

A service connected to the antenna (or cell tower) preferably uses historical information and artificial intelligence interrogation of MS travels to determine fields **1100h** and **1100i**. Block **236** continues to block **238** where parameters are prepared for passing to FIG. 2F processing invoked at block **240**. Parameters are set for: WDRREF=a reference or

US 10,477,994 B2

43

pointer to the WDR; DELETEQ=FIG. 2D location queue discard processing; and SUPER=FIG. 2D supervisory notification processing. Thereafter, block 240 invokes FIG. 2F processing and FIG. 2D processing terminates at block 242. FIG. 2F processing will insert to queue 22 so this MS knows at least its own whereabouts whenever possible. A single data instance embodiment of WDR queue 22 will cause FIG. 2F to update the single record of WDR information for being current upon exit from block 240 (this is true for all flowchart blocks invoking FIG. 2F processing).

With reference now to FIG. 2F, depicted is a flowchart for describing a preferred embodiment of a procedure for inserting a Whereabouts Data Record (WDR) 1100 to MS WDR queue 22. Appropriate semaphores are used for variables which can be accessed simultaneously by another thread other than the caller. With reference now to FIG. 2F, procedure processing starts at block 270 and continues to block 272 where parameters passed from the invoking block of processing, for example block 240, are determined. The variable WDRREF is set by the caller to a reference or pointer to the WDR so subsequent blocks of FIG. 2F can access the WDR. The variable DELETEQ is set by the caller so that block 292 knows how to discard obsolete location queue entries. The DELETEQ variable can be a multi-field record (or reference thereof) for how to prune. The variable SUPER is set by the caller so that block 294 knows under what condition(s), and which data, to contact a supervisory service. The SUPER variable can be a multi-field record (or reference thereof) for instruction.

Block 272 continues to block 274 where the DLMV (see FIG. 12 and later discussions for DLMV (DLM role(s) List Variable)), or ILMV (see FIG. 12 and later discussions for ILMV (ILM role(s) List Variable)), is checked for an enabled role matching the WDR for insertion (e.g. DLM: location technology field 1100e (technology and originator indicator) when MS ID=this MS; ILM: DLM or ILM indicator when MS ID not this MS). If no corresponding DLMV/ILMV role is enabled for the WDR to insert, then processing continues to block 294 (the WDR is not inserted to queue 22). If the ILMV/DLMV role for the WDR is enabled, then processing continues to block 276 where the confidence of the WDR 1100 is validated prior to insertion. An alternate embodiment to FIG. 2F will not have block 274 (i.e. block 272 continues directly to block 276) since appropriate DLM and/or ILM processing may be terminated anyway when DLM/ILM role(s) are disabled (see FIG. 14A/B).

If block 276 determines the data to be inserted is not of acceptable confidence (e.g. field 1100d < confidence floor value (see FIG. 14A/B)), then processing continues to block 294 described below. If block 276 determines the data to be inserted is of acceptable confidence (e.g. field 1100d > 70), then processing continues to block 278 for checking the intent of the WDR insertion.

If block 278 determines the WDR for insert is a WDR describing whereabouts for this MS (i.e. MS ID matching MS of FIG. 2F processing (DLM: FIGS. 2A through 9B, or ILM: FIG. 26A/B)), then processing continues to block 280. If block 278 determines the WDR for insert is from a remote ILM or DLM (i.e. MS ID does not match MS of FIG. 2F processing), then processing continues to block 290. Block 280 peeks the WDR queue 22 for the most recent highest confidence entry for this MS whereabouts by searching queue 22 for: the MS ID field 1100a matching the MS ID of FIG. 2F processing, and a confidence field 1100d greater than or equal to the confidence floor value, and a most recent date/time stamp field 1100b. Thereafter, if block 282 deter-

44

mines one was found, then processing continues to block 284, otherwise processing continues to block 286 where a Last Whereabouts date/Time stamp (LWT) variable is set to field 1100b of the WDR for insert (e.g. first MS whereabouts WDR), and processing continues to block 288.

If block 284 determines the WDR for insertion has significantly moved (i.e. using a movement tolerance configuration (e.g. 3 meters) with fields 1100c of the WDR for insert and the WDR peeked at block 280), then block 286 sets the LWT (Last Whereabouts date/Time stamp) variable (with appropriate semaphore) to field 1100b of the WDR for insert, and processing continues to block 288, otherwise processing continues directly to block 288 (thereby keeping the LWT as its last setting). The LWT is to hold the most recent date/time stamp of when the MS significantly moved as defined by a movement tolerance. The movement tolerance can be system defined or configured, or user configured in FIG. 14 by an option for configuration detected at block 1408, and then using the Configure Value procedure of FIG. 18 (like confidence floor value configuration).

Block 288 accesses the DLMV and updates it with a new DLM role if there is not one present for it. This ensures a correct list of DLMV roles are available for configuration by FIG. 14. Preferably, by default an unanticipated DLMV role is enabled (helps inform the user of its availability). Likewise in another embodiment, ILMV roles can be similarly updated, in particular if a more granulated list embodiment is maintained to the ILMV, or if unanticipated results help to identify another configurable role. By default, block 274 should allow unanticipated roles to continue with WDR insertion processing, and then block 288 can add the role, enable it, and a user can decide what to do with it in configuration (FIG. 14A/B).

Thereafter, the WDR 1100 is inserted to the WDR queue 22 at block 290, block 292 discards any obsolete records from the queue as directed by the caller (invoker), and processing continues to block 294. The WDR queue 22 preferably contains a list of historically MS maintained Whereabouts Data Records (WDRs) as the MS travels. When the MS needs its own location, for example from an application access, or to help locate an ILM, the queue is accessed for returning the WDR with the highest confidence value (field 1100d) in the most recent time (field 1100b) for the MS (field 1100a). Block 292 preferably discards by using fields 1100b and 1100d relative to other WDRs. The queue should not be allowed to get too large. This will affect memory (or storage) utilization at the MS as well as timeliness in accessing a sought queue entry. Block 292 also preferably discards WDRs from queue 22 by moving selected WDRs to LBX History 30.

As described above, queue interfaces assume an implicit semaphore for properly accessing queue 22. There may be ILMs requesting to be located, or local applications of the MS may request to access the MS whereabouts. Executable thread(s) at the MS can access the queue in a thread-safe manner for responding to those requests. The MS may also have multiple threads of processing for managing whereabouts information from DLMs, ILMs, or stationary location services. The more concurrently executable threads available to the MS, the better the MS is able to locate itself and respond to others (e.g. MSs). There can be many location systems and methods used to keeping a MS informed of its own whereabouts during travel. While the preferred embodiment is to maximize thread availability, the obvious minimum requirement is to have at least 1 executable thread available to the MS. As described above, in operating system environments without proper queue interfaces, queue access